Universität Passau
Fakultät für Informatik und Mathematik

Bachelorarbeit

Text-Fehlerkorrektur durch word2vec mit numerischen Klassifikations-Schranken

Maria Kober

7. Januar 2016

Bachelorarbeit
am Lehrstuhl für Medieninformatik
der Fakultät für Informatik und Mathematik
der Universität Passau

Erstgutachter: Prof. Dr. Michael Granitzer
Betreuer: Johannes Jurgovsky

Kurzfassung

In dieser Arbeit werden automatische Methoden zur Identifizierung von Digitalisierungsfehlern in historischen, mit OCR digitalisierten Zeitungsartikeln untersucht. Durch die Identifizierung der fehlerhaften Wörter soll der manuelle Korrekturaufwand solcher Texte erleichtert werden.

Zur Erkennung von Fehlern werden mehrere binäre, Schwellwert-basierte Klassifikatoren verwendet. Die Klassifikatoren basieren auf dem Word2Vec-Modell oder arbeiten mit den Word2Vec-Wortvektoren.

Zur Evaluierung der Klassifikatoren werden zwei Word2Vec-Modelle berechnet: eines mit einem domänenspezifischen und eines mit einem domänenunabhängigen Trainingskorpus.

Die durchgeführten Experimente zeigen, dass die getesteten Klassifikatoren nicht zur Identifizierung von fehlerhaften Wörtern geeignet sind. Das domänenunabhängige Modell enthält mehr Wörter im Vokabular als das domänenspezifische Modell und erzielt dadurch bessere Ergebnisse. Ansonsten haben sich bei der Verwendung eines domänenspezifischen und eines domänenunabhängigen Modells keine nennenswerten Unterschiede ergeben.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Motivation

Das Thema dieser Arbeit die Fehlererkennung und Fehlerkorrektur von digitalen Texten. Spezifisch wird die Fehlererkennung an Texten untersucht, die mit einem OCR[1]-Programm digitalisiert wurden. Die Digitalisierung ist im Allgemeinen nicht fehlerfrei. Der digitalisierte Text muss manuell durchgesehen und gegebenenfalls korrigiert werden.

Ziel dieser Arbeit ist es, bestimmte automatische Methoden zu untersuchen, mit denen der Korrekturaufwand reduziert werden kann. Das heißt, die Methoden erkennen fehlerhafte Wörter und der User kann diese gezielt korrigieren. Falls diese Methoden eine sinnvolle Arbeitsreduktion ermöglichen, soll die Entwicklung einer Software möglich sein, welche den digitalisierten Text durchgeht und alle Wörter anzeigt, die wahrscheinlich fehlerhaft sind. Dadurch müsste dann nicht mehr der gesamte digitalisierte Text Korrektur gelesen werden. Es würde ausreichen, einzelne Wörter manuell auf ihre Richtigkeit zu überprüfen.

Konkret werden die Methoden auf die Donau-Zeitung vom Anfang des 20. Jahrhunderts angewandt und getestet. Diese in Frakturschrift gedruckte Zeitung wurde im Rahmen des Projekts *Ostbayern im I. Weltkrieg. Die „Donau-Zeitung“ digital*[2] digitalisiert. Da diese Digitalisierung nicht fehlerfrei war, mussten die Texte von Hand korrigiert werden. Aufgrund der großen Textmenge konnte jedoch nur ein Teil der digitalisierten Zeitungsartikel korrigiert und veröffentlicht werden. Für Beispiele einer korrekten und unkorrekten Digitalisierung, siehe Abbildung 1.1.

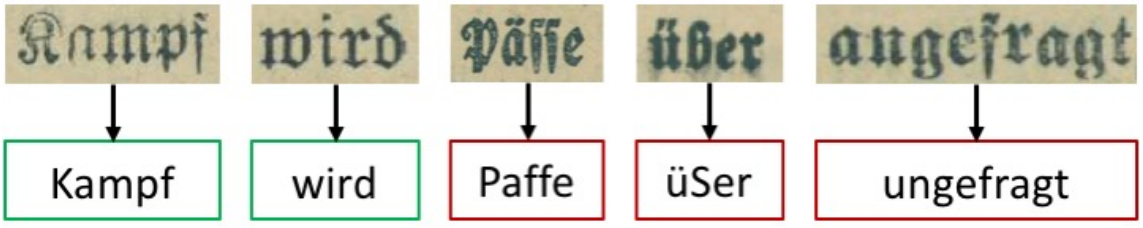

Abbildung 1.1: Zeitungswörter mit ihrer jeweiligen Digitalisierung. Links: Die Wörter *Kampf* und *wird* wurden korrekt digitalisiert. Rechts: Die Wörter *Pässe*, *über* und *angefragt* wurden falsch digitalisiert.

Mit den bereits von Hand korrigierten Texten wird in dieser Arbeit u.a. überprüft, wie zuverlässig die Methoden zur Fehlererkennung arbeiten.

[1] OCR ist die Abkürzung für Optical Character Recognition, zu Deutsch Optische Zeichenerkennung.

[2] `http://www.phil.uni-passau.de/die-fakultaet/lehrstuehle-professuren/rehbein/forschung/donau-zeitung-digital.html` – zuletzt abgerufen am 24. 06. 2015 und `http://dz1914.uni-passau.de/` – zuletzt abgerufen am: 24. 09. 2015

1.1 Fragestellung

In dieser Arbeit sollen 3 Fragen beantwortet werden:

1. Eignen sich das Word2Vec-Modell und die Word2Vec-Wortvektoren für eine Schwellwert-basierte Fehlererkennung?
Die Eingangs dargelegte Problemstellung soll mit der Methode *Word2Vec* bearbeitet werden. Daraus ergibt sich die naheliegende Frage, ob das resultierende Word2Vec-Modell und die Word2Vec-Wortvektoren zu einer Schwellwert-basierten Fehlererkennung geeignet sind.

2. Wie verhalten sich unterschiedliche Schwellwert-basierte Klassifikatoren bei verschiedenen Schwellwerten?
Hier soll ermittelt werden, welche Schranke gesetzt werden soll, um ein Wort als *korrekt* oder *fehlerhaft* zu klassifizieren. Muss ein Wort korrigiert werden, wenn die Methode eine 50%ige Wahrscheinlichkeit angibt, dass das Wort falsch ist? Oder erst bei 75%? Oder einer ganz anderen Klassifikations-Schranke?

Dieser Schwellwert soll anhand der vorhandenen Daten bestmöglich bestimmt werden. Das heißt, die Software soll möglichst alle falsche Wörter sicher finden, und so viele korrekte Wörter wie möglich als korrekt anzeigen.

3. Lassen sich mit domänenspezifischen Wortvektoren bessere Ergebnisse erzielen als mit domänenunabhängigen Wortvektoren?
Hier soll untersucht werden, wie sich die Genauigkeit bzw. Zuverlässigkeit der Methoden verändert, wenn die Wortvektoren aus unterschiedlichen Trainingsdaten generiert werden. Konkret wird dies an zwei unterschiedlichen Textkorpora untersucht. Der erste Textkorpus besteht aus den Texten der deutschen Wikipedia. Diese Texte stammen aus der heutigen Zeit und sind domänenunabhängig. Der zweite Textkorpus besteht aus einem Teil der korrigierten Zeitungsartikel der *Donau-Zeitung*. Dieser ist deutlich kleiner als der Wikipedia-Textkorpus und domänenspzifisch.

1.2 Aufbau der Arbeit

Im zweiten Kapitel werden theoretische Grundlagen erläutert. Zunächst wird auf Methoden zur automatisierten Text-Fehlererkennung eingegangen. Danach folgt ein Abschnitt über die Fehlerkorrektur im OCR Postprocessing. Folgend wird ein Überblick über Wortvektoren gegeben. Im letzten Abschnitt wird erklärt, was Word2Vec ist und wie es funktioniert.

Im dritten Kapitel werden mehrere binäre, Schwellwert-basierte Klassifikatoren vorgestellt.

Im vierten Kapitel werden die Rahmenbedingungen zu den durchgeführten Experimenten erläutert. Dies schließt die verwendeten Textkorpora, verwendete Parameter, Validierungsdaten, technische Rahmenbedingungen und die Vorverarbeitung der Texte ein.

Im fünften Kapitel werden die Klassifikatoren aus Kapitel 3 evaluiert. Im ersten Teil werden einige ausgewählte Ergebnisse aus den durchgeführten Experimenten vorgestellt. Im zweiten Teil werden die Ergebnisse diskutiert und analysiert.

Das sechste Kapitel beinhaltet eine kurze Zusammenfassung dieser Arbeit.

2 Grundlagen

Dieses Kapitel ist in vier Abschnitte gegliedert. Der erste Abschnitt gibt einen Überblick über Methoden zur automatisierten Fehlererkennung in Texten. Der zweite Abschnitt geht auf Methoden zur Fehlerkorrektur im OCR Postprocessing ein. Im dritten Abschnitt werden Wortvektoren erklärt und im vierten Abschnitt wird die Funktionsweise von Word2Vec erläutert.

2.1 Automatisierte Text-Fehlererkennung

In der Literatur werden meist 2 Arten von fehlerhaften Wörtern unterschieden, so beispielsweise von Kukich in [1] und Ahmed et al. in [2]:

1. Wörter, die keinen Sinn ergeben und nicht existieren.
2. Wörter, die existierende Wörter darstellen, die in ihrem Kontext jedoch fehlerhaft sind.

2.1.1 Fehlererkennung ohne Wortkontext

Im Folgenden wird ein knapper Überblick über grundlegende Techniken gegeben, die Fehler ohne die Verwendung eines Wortkontexts erkennen.

Kukich gibt in [1] einen Überblick über Techniken, die zur Fehlererkennung und -korrektur verwendet werden. Obwohl die Arbeit schon älter ist, werden viele der Techniken bis heute erfolgreich angewandt. Kukich gibt zwei grundlegende Vorgehensweisen an, um fehlerhafte Wörter der ersten Kategorie zu finden: n-gramme und Wörterbuchsuche.

Ein n-gramm besteht aus n aufeinander folgenden Buchstaben. Meist ist n = 1, 2 oder 3. Man spricht dann von einem Monogramm, Bigramm oder Trigramm. Zunächst werden existierende und valide n-gramme ermittelt und in einer geeigneten Datenstruktur gespeichert. Ein Wort wird auf seine Korrektheit geprüft, indem jedes n-gramm aus den Buchstaben des Wortes mit der Liste der validen n-grammen verglichen wird. Kommt das beobachtete n-gramm dort nicht vor, ist das Wort wahrscheinlich fehlerhaft. Dieses Verfahren wird u.a. von OCR-Programmen verwendet. Ahmed et al. beziehen in [2] zusätzlich die Position der n-gramme innerhalb eines Wortes mit ein.

Die Wörterbuchsuche ist im Ansatz sehr simpel. Das Wörterbuch besteht aus einer Liste aus validen Wörtern. Die Liste kann von Hand, durch ein bestehendes Lexikon oder durch Extraktion aus einem Textkorpus erstellt werden. Ein Wort wird als fehlerhaft eingestuft, wenn es nicht im Wörterbuch zu finden ist.

Der Zugriff auf die Wörterbucheinträge kann durch verschiedene Verfahren, z.B. Hashing, optimiert werden.

Ein der Wörterbuchsuche ähnlicher Ansatz wird beispielsweise von Hassan et al. in [3] vorgestellt. Der Ansatz arbeitet mit endlichen Automaten. Jedes Wort im Wörterbuch und im Eingabestring wird jeweils durch einen Pfad in einem endlichen Automat dargestellt. Gibt es zum Eingabestring-Pfad keinen Pfad im endlichen Automaten des Wörterbuchs, wird das Wort als fehlerhaft angenommen.

2.1.2 Fehlererkennung mit Wort- oder Satzkontext

Die im Folgenden vorgestellten Arbeiten und Techniken verwenden den Wort- oder Satzkontext eines Wortes, um seine Korrektheit zu prüfen. Der Wortkontext kann auf verschiedene Zusammenhänge untersucht werden: semantische, syntaktische und grammatikalische, sowie auf weitere Merkmale wie die innere Struktur eines Textes. Mit diesen Techniken sollen vor allem Wörter aus der zweiten Fehlerkategorie, existierende Wörter, erkannt werden [1].

Statistische Sprachmodelle werden häufig zur Fehlererkennung und -korrektur verwendet und teils mit anderen Techniken kombiniert. Bei einem statistischen Modell wird ermittelt, wie wahrscheinlich ein Wort oder Satzteil ist, gegeben sein Kontext.

Richter et al. beschreiben in [4] ein Fehlerkorrektur-Programm, das auf mehreren sprachunabhängigen statistischen Modellen aufbaut.

Eine statistische Technik ist die Verwendung eines n-gramms auf Wortebene statt auf Buchstabenebene [1]. Diese Technik wurde verfeinert und optimiert, beispielsweise von Samanta et al. in [5], die Bi- und Trigramme gemeinsam verwenden. Whitelaw et al. verwenden in [6] das World Wide Web, um n-gramme zu erstellen sowie Fehler zu erkennen und zu korrigieren. n-gramme werden auch mit anderen Techniken kombiniert, so beispielsweise von Golding & Schabes in [7]. In ihrer Arbeit benutzen sie Trigramme und kombinieren diese mit einem Bayes-Klassifikator. Das Rechtschreibprogramm *Hunspell*[3], das bspw. von *LibreOffice* und *InDesign* verwendet wird, nutzt u.a. n-gramme zur Fehlerkorrektur und -erkennung.

Die Grammatik und Syntax einer Sprache kann benutzt werden, um fehlerhafte Wörter zu finden. Bick nutzt in [8] dänische Grammatikregeln, um Fehler in einem Text zu finden. Hermet et al. verwenden in [9] Informationen aus dem World Wide Web, um eine falsche Verwendung von Präpositionen zu erkennen und zu korrigieren.

Hirst & Budanitsky untersuchen in [10] die Fehlererkennung und -korrektur anhand der semantischen Ähnlichkeit von Wörtern in einem Wörterbuch.

Methoden für maschinelles Lernen werden verwendet, um Fehler zu erkennen und zu korrigieren. Eine Methode ist die Anwendung eines Bayes-Klassifikators, wie bspw. von Golding in [11]. Golding & Roth wenden in [12] den Winnow-Algorithmus an, um Fehler abhängig vom Wortkontext zu finden. Sjöbergh & Knutsson untersuchen in [13] eine Methode, welche Fehlererkennung völlig unüberwacht erlernt.

[3] `http://hunspell.sourceforge.net/` – zuletzt aufgerufen am 27. 12. 2015

Latent Semantic Indexing, im englischen Originalbegriff Latent Semantic Analysis, wurde zur Fehlererkennung im Wortkontext herangezogen. Eine der ersten Arbeiten, die sich damit beschäftigt, wurde von Jones & Martin veröffentlicht, siehe [14].

Das Noisy Channel Modell wurde in mehreren Arbeiten zur Fehlererkennung oder in Kombination mit anderen Techniken verwendet. So beispielsweise von Brill & Moore in [15] und von Whitelaw et al. in [6], hier in Kombination mit n-grammen.

Insgesamt scheint jedoch noch kein optimaler Ansatz gefunden worden zu sein, mit dem Fehler, die in einem existierenden Wort resultieren, in einem Text gefunden werden können. Es gibt sehr viele Ansätze und Kombinationen von mehreren Ansätzen, von denen nur ein kleiner Teil in diesem Abschnitt abgebildet wurde.

2.1.3 Einordnung dieser Arbeit

Word2Vec ist ein Wortkontext-basierter Ansatz, der ein statistisches Sprachmodell berechnet. Zur Modellierung wird ein neuronales Netz verwendet.
Die Methoden, die in dieser Arbeit zur Fehlererkennung angewandt werden, basieren ebenfalls auf dem Wortkontext eines Wortes. Sie nutzen teilweise die berechneten Wahrscheinlichkeiten des Sprachmodells.
Die Herausforderung besteht im Folgenden darin, korrekte Wörter zu finden, die in ihrem Wortkontext falsch sind.

Das Word2Vec-Modell erstellt ein Wörterbuch. Dieses Wörterbuch wird wie in Abschnitt 2.1.1 beschrieben genutzt, um nicht existierende Wörter zu erkennen. Diese Arbeit beschäftigt sich nicht mit Wörterbuch-Techniken, sondern benutzt lediglich ein Wörterbuch.

2.2 Fehlerkorrektur im OCR Postprocessing

Die Validierungstexte und ein Trainingskorpus, die in dieser Arbeit verwendet werden, wurden mit einem OCR-Programm digitalisiert. Zunächst wird kurz auf Probleme bei der Digitalisierung historischer Texte eingegangen. Im Anschluss wird erläutert, welche automatischen Techniken nach dem eigentlichen Digitalisierungsprozess zur Fehlererkennung und -korrektur angewandt werden können.

2.2.1 Probleme bei der Digitalisierung historischer Texte

Inzwischen können OCR-Programme einen Text bis zu 99% korrekt digitalisieren. Dies gilt vor allem für Dokumente mit guter Qualität und hohem Kontrast zwischen Buchstaben und Papier [16].

Bei historischen Texten, bspw. Zeitungen, ist der Druck im Allgemeinen nicht sehr hochwertig. Es kann sein, dass der Text von der Rückseite durchschlägt oder dass die Zeitung beschädigt oder verschmutzt ist [17]. Ein hoher Kontrast zwischen Papier und Buchstaben ist dort ebenfalls häufig nicht mehr gegeben. Dazu kommt meist eine nicht mehr zeitgemäße Schrift, wie zum Beispiel die Frakturschrift [16].

Klijn hat 2008 mehrere kommerzielle OCR-Programme mit Zeitungen aus dem beginnenden 20. Jahrhundert getestet und verglichen, siehe [17]. Dabei wurden bis zu 32% des Zeitungstextes fehlerhaft digitalisiert.

2.2.2 Automatisierte Fehlerkorrektur

Kukich stellt in [1] fest, dass die häufigste Fehlerquelle von OCR-Programmen die Verwechslung ähnlich aussehender Buchstaben ist. Beispielsweise wird ein `D` zu einem `O` oder ein `S` zu einer `5`.

Eine Möglichkeit, um Fehler nach der Digitalisierung zu erkennen und zu beheben, sind n-gramme auf Buchstabenebene. In Abschnitt 2.1 wurde die Fehlererkennung mit n-grammen erläutert.
Um ein Wort zu korrigieren, gibt es mehrere Möglichkeiten.

Riseman & Hanson stellen in [18] eine Methode vor, mit der ein Fehler innerhalb eines n-gramms korrigiert werden kann. Zunächst werden lagerichtige n-gramme erstellt. Das heißt, zusätzlich zu den Buchstaben des n-gramms werden die möglichen Positionen in einem Wort ermittelt. Wird ein Wort durch mehrere n-gramme als fehlerhaft identifiziert, kann die Position des Fehlers ermittelt werden. Der fehlerhafte Buchstabe wird durch den Buchstaben in einem n-gramm korrigiert, das mit den umgebenden Buchstaben an der Position des Fehlers valide ist. Dieses Vorgehen funktioniert nicht, wenn mehrere n-gramme an der Fehlerposition valide sind.

Holley schlägt in [16] die Verwendung einer Wahrheitsmatrix zur OCR-Fehlerkorrektur vor. In der Wahrheitsmatrix sind häufig verwechselte Buchstaben enthalten. Wird ein Buchstabe als fehlerhaft identifiziert, kann er mithilfe der Wahrheitsmatrix mit dem korrekten Buchstaben ausgetauscht werden. Die Identifikation des Fehlers erfolgt mit n-grammen.

Tong & Evans kombinieren in [19] die Verwendung von n-grammen mit einer Wahrheitsmatrix und einem statistischen Sprachmodell für Bigramme.

Eine weitere Technik, um fehlerhafte Wörter zu identifizieren, ist die Wörterbuchsuche [20]. In Abschnitt 2.1 wurde diese Art der Fehlererkennung erläutert. Zur Ermittlung eines korrekten Wortes kann bspw. das Wort mit der kürzesten Distanz zum falschen Wort ermittelt werden. Als Distanzmaß kann die Levenshtein-Distanz verwendet werden [1].

Zur Fehlerkorrektur können statistische Sprachmodelle und die grammatikalischen und syntaktischen Regeln einer Sprache herangezogen werden [20].

Niwa et al. verwenden in [21] ein Wörterbuch und grammatikalische Regeln, um Fehler zu erkennen und zu korrigieren. Zusätzlich werden syntaktische Informationen, wie der Teil eines Satzes, in dem sich das Wort befindet, zur Korrektur benutzt.

Tong et al. verwenden in [22] ein Wörterbuch und ein statistisches Sprachmodell, um Fehler zu finden und mögliche korrekte Wörter zu ermitteln. Das Wort, durch welches das fehlerhafte Wort ersetzt wird, wird durch die Maximum-Likelihood-Funktion bestimmt.

Hull erstellt in [23] ein statistisches Sprachmodell und bezieht die syntaktische Information, zu welcher Struktur das Wort innerhalb eines Satzes gehört, mit ein.

Guyon & Pereira verwenden in [24] probabilistische endliche Automaten, um ein Wort zu korrigieren.

Das Open Source Programm *OCRopus*[4] arbeitet mit statistischen Sprachmodellen. Das schließt Grammatik, n-gramme auf Wort- und Buchstabenebene und Wörterbücher mit ein [25].

Verschiedene Arten von neuronalen Netzen werden verwendet. Diese werden nicht nur im Postprocessing, sondern auch während der eigentlichen Digitalisierung eingesetzt.

Breuel et al. verwenden in [26] ein Long-Short Term Memory Network für OCR. In ihre Arbeit beziehen sie speziell Texte in Frakturschrift mit ein.

Bengio & Le Cun nutzen in [27] ein Convolutional Neural Network, grammatikalische Regeln und ein Hidden Markov Model im OCR Postprocessing. In ihrer Arbeit geht es um die Erkennung von Wörtern, die aktiv vom User eingegeben werden.

Wie aus den vorangegangenen Abschnitten ersichtlich ist, gibt es beim OCR Postprocessing viele Ansätze und Methoden, um Fehler zu korrigieren. Diese Ansätze überschneiden sich z. T. mit Methoden zur Korrektur von beliebigen, nicht mit OCR digitalisierten Texten.

2.3 Verteilte Wortvektoren

Verteilte Wortvektoren sind Vektoren in $\mathbb{R}^N$. Die Wortvektoren werden i. A. durch ein neuronales Netz erzeugt. Dieses kann mit einem (statistischen) Sprachmodell kombiniert sein [28].

Es wird davon ausgegangen, dass sich alle Wörter einer Sprache als Punkte in einem N-dimensionalen Vektorraum darstellen lassen. Jede Koordinate des Vektors soll eine Eigenschaft des Wortes repräsentieren [29]. Diese Eigenschaft kann beispielsweise semantisch oder grammatikalisch sein [30].

Turney & Pantel geben in [31] einen Überblick über verschiedene Anwendungsgebiete von Wortvektoren.
Ein Anwendungsgebiet ist die Erstellung einer Wort-Kontext-Matrix. Hier wird davon ausgegangen, dass die Bedeutung eines Wortes durch seinen Kontext bestimmt wird und dass Wörter mit ähnlichem Kontext meist eine ähnliche Bedeutung haben. Mit dieser Vektordarstellung können Wörter in vorgegebene Kategorien einordnet werden oder korrigiert werden, wenn sie in ihrem Kontext falsch sind. Turney & Littman haben sich in [32] mit der Kategorisierung von Wörtern und Texten beschäftigt. Sie stellen dazu zwei Ansätze vor – einer davon verwendet Latent Semantic Indexing zusammen mit Wortvektoren.
Eine weitere Anwendung ist die Bestimmung von Ähnlichkeiten zwischen Wörtern

[4] `https://github.com/tmbdev/ocropy` – zuletzt aufgerufen am 29. 12. 2015

und Wortpaaren. Damit können u.a. semantische oder syntaktische Relationen zwischen Wörtern und Wortpaaren abgebildet werden. Diese Eigenschaften können beispielsweise bei einer relationalen Suche verwendet werden oder um Analogien zu finden.

Collobert et al. beschreiben in [33] einen Ansatz, um problemunabhängig verteilte Wortvektoren zu lernen. Problemunabhängig heißt, dass das neuronale Netz und folglich die Vektoren nicht speziell auf eine Anwendung hin trainiert werden.
In ihrer Arbeit nennen sie mehrere Informationsquellen, mit denen ein Wortvektor trainiert werden kann. Zum Einen kann ein Wortkontext der Größe $k \in \mathbb{N}$ verwendet werden, der das betrachtete Wort umgibt. Zum Anderen kann der gesamte Satz verwendet werden. Dies ist zum Beispiel sinnvoll, wenn die Syntax eines getrennten Verbs erfasst werden soll.

Eine der ersten Verwendungen von Wortvektoren, die automatisiert erlernt werden, wird in einigen Arbeiten Hinton et al. zugeschrieben, bspw. in [34], [29] und [30]. Es werden mehrere Arbeiten zitiert: In [35] wird ein Rückpropagierungs-Algorithmus vorgestellt, mit dem ein künstliches neuronales Netz Vektorrepräsentationen automatisiert lernen kann. In [36] wird genauer auf die verteilte Darstellung der gelernten Repräsentationen und der Konzepte dahinter eingegangen. In [37] wird auf mehrere Arten von neuronalen Netzen, Lernalgorithmen und die Repräsentation von Informationen durch diese Netze eingegangen.

Eine Arbeit von Bengio et al. wird ebenfalls mehrfach genannt, beispielsweise in [30] und [38]. In [29] kombinieren Bengio et al verteilte Wortvektoren mit einem statistischen Sprachmodell. Damit ist es möglich, dass semantisch ähnlichen Wörtern eine ähnliche Wahrscheinlichkeit in einem gleichen Kontext zugeordnet wird. Dazu ist es nicht notwendig, dass jedes dieser Wörter während der Trainingsphase des neuronalen Netzes im selben Kontext auftaucht. Es genügt, dass ein Wort im Kontext vorkommt, und dieses Vorkommen wird dann auf semantisch ähnliche Wörter übertragen.

2.4 Word2Vec

Word2Vec ist ein Modell zur Berechnung von Wortvektoren. Es wurde 2013 von Milkov et al in [38] vorgestellt.

Zur Ermittlung der Wortvektoren wird ein möglichst großer Trainingskorpus aus Texten benötigt. Die Position der Wortvektoren im Vektorraum wird durch die Wortkontext-Kookkurenz des Wortes ermittelt. Der Wortkontext bezeichnet hier die k Wörter links und die k Wörter rechts von einem Wort.

Zur Berechnung der Wortvektoren wurden in [38] zwei Modelle vorgestellt: Continuous Skip-Gram und Continuous Bag-of-Words, kurz CBOW. Die Vorgehensweise, wie die Wortvektoren ermittelt werden, ist bei beiden Modellen verschieden. Konkret sind die Ansätze genau gegensätzlich.
In dieser Arbeit wird mit CBOW gearbeitet. Daher wird auf die Berechnungsweise von Continuous Skip-Gram im Folgenden nicht eingegangen.

Rong erklärt in [39] sehr detailliert, wie CBOW und Skip-Gram trainiert werden. Die folgende Ausführung ist diesem Paper und den ursprünglichen Ausführungen von Mikolov et al, siehe [38], entnommen.

Um den oben beschriebenen Vektorraum zu erzeugen, verwendet das CBOW-Modell ein neuronales Netz. Die Netz-Architektur ist in Abbildung 2.1 abgebildet.

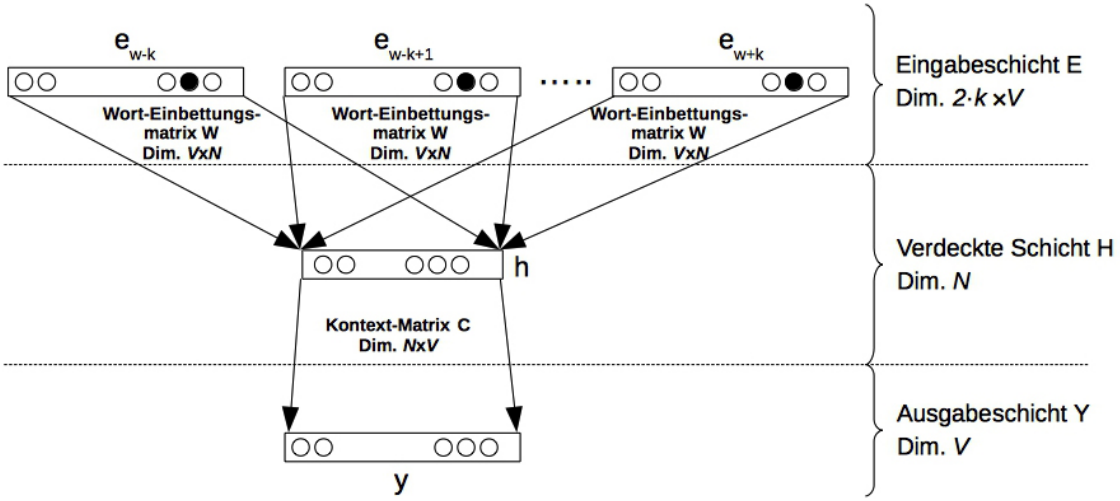

Abbildung 2.1: Netz-Architektur des Continuous Bag-of-Words Modells. Zukünftige und vergangene Kontextwörter werden auf ihren Wortvektor projiziert und dann gemittelt. Dieser mittlere Wortvektor befindet sich in der verdeckten Schicht H. Danach wird für jedes Wort im Vokabular eine bedingte Wahrscheinlichkeit gegeben die Kontextwörter berechnet. Diese Wahrscheinlichkeiten werden in der Ausgabeschicht Y angegeben.

Zunächst wird aus dem gegebenen Trainingskorpus ein Vokabular extrahiert. Sei V im Folgenden die Anzahl der Wörter im Vokabular. Sei w_i das Wort mit Index i im Vokabular, $i \in \{1, \ldots, V\}$. Sei w_x das Wort, mit dem das neuronale Netzwerk in Trainingsschritt x traininert wird, $x \in \{1, \ldots, |Trainingskorpus|\}$. Der Wortkontext besteht aus den k Wörtern, die im Trainingstext links von w_x stehen und den k Wörtern rechts von w_x. Seien die Kontextwörter im Folgenden mit $w_{x-k}, \ldots, w_{x-1}, w_{x+1}, \ldots, w_{x+k}$ bezeichnet.

In jedem Trainingsschritt wird eine Wahrscheinlichkeitsverteilung über das gesamte Vokabular berechnet. Für jedes Wort im Vokabular wird die folgende Wahrscheinlichkeit berechnet:

$$P(w_l | w_{x-k}, \ldots, w_{x-1}, w_{x+1}, \ldots, w_{x+k}) \in [0, 1], l \in \{1, \ldots, V\}$$

Zur Berechnung der Wahrscheinlichkeit sind mehrere Schritte nötig.

Die Eingabeschicht E besteht aus einer $2 \cdot k \times V$-Matrix. Die Zeilen der Matrix sind 1-aus-V-kodiert. In jeder Zeile ist eines der Kontextwörter von w_x aktiviert.

Die Wort-Einbettungs-Matrix W verbindet die Eingabeschicht E mit der verdeckten Schicht H. W ist eine $V \times N$-Matrix. Jede Zeile der Matrix repräsentiert einen

Wortvektor $v_i \in \mathbb{R}^N$, $i \in \{1, \ldots, V\}$. v_i ist der Wortvektor von Wort w_i.
Jede Zeile von E wird mit W multipliziert. Dadurch wird jedes Kontextwort von w_x auf seinen entsprechenden Wortvektor abgebildet; man erhält die Wortvektoren $v_{x-k}, \ldots, v_{x-1}, v_{x+1}, \ldots, v_{x+k}$.

Die verdeckte Schicht H besteht aus einem N-dimensionalen Vektor. Die Ausgabe der verdeckten Schicht ist der mittlere Vektor $h \in \mathbb{R}^N$ der Kontextwörter:

$$h = \frac{1}{2 \cdot k} \cdot (v_{x-k} + \cdots + v_{x-1} + v_{x+1} + \cdots + v_{x+k})$$

Die Kontext-Matrix C verbindet die verdeckte Schicht mit der Ausgabeschicht Y. C ist eine $N \times V$-Matrix. Jede Spalte der Matrix repräsentiert einen Kontextvektor $c_i \in \mathbb{R}^N$, $i \in \{1, \ldots, V\}$. c_i ist der Kontextvektor zu Wort w_i.

Die Ausgabeschicht Y hat Dimension V. Sie enthält für alle Wörter im Vokabular die bedingte Wahrscheinlichkeit $p_i \in [0, 1]$, $i \in \{1, \ldots, V\}$, von Wort w_i gegeben die Kontextwörter in Trainingsschritt x. Die bedingte Wahrscheinlichkeit für w_i wird wie folgt berechnet:

$$\begin{aligned} p_i &= P(w_i | w_{x-k}, \ldots, w_{x-1}, w_{x+1}, \ldots, w_{x+k}) \\ &= \frac{e^{h \cdot c_i}}{\sum_{j=1}^{V} e^{h \cdot c_j}} \in [0, 1] \end{aligned}$$

Die Vorhersage des Netzes, die in Y angezeigten Wahrscheinlichkeiten, werden mit den realen Begebenheiten im Trainingskorpus verglichen. Abweichungen werden mit einer Fehlerfunktion gemessen; laut Rong handelt es sich bei dieser Funktion um eine Spezialform der Kreuzentropie. Die Vektoren in W und C werden beispielsweise durch Rückpropagierung so abgeändert, dass die in dem Trainingsschritt gegebene Wort-Wortkontext-Kombination eine entsprechend hohe Wahrscheinlichkeit erzielt. Ziel der Anpassung ist die Minimierung des Vorhersagefehlers, also der Fehlerfunktion.

Durch das Trainieren des Netzwerks werden die zufällig initialisierten Wortvektoren an ihren richtigen Platz verschoben. Damit ist nicht ein konkreter Punkt im Vektorraum gemeint, sondern das Verhältnis zu anderen Wörtern im Vektorraum.

In [38] wurde gezeigt, dass durch die Vektorrepräsentation bestimmte Ähnlichkeitsmerkmale der Wörter bestehen bleiben. Damit konnten beispielsweise Fragen nach der Ähnlichkeit zwischen Wörtern und Wortpaaren automatisch beantwortet werden.

Zwei Beispiele für eine solche Ähnlichkeitsbeziehung sind die Komparativ-Relation und die Antonymie-Relation. Als Beispielwörter seien hier die Wörter `groß`, `größer`, `klein` und `kleiner` gegeben. Die Relationen zwischen den jeweiligen Wortpaaren sind in Abbildung 2.2 im zweidimensionalen Raum veranschaulicht.
Stellt man die Frage, welches Wort zu `klein` genauso ist wie `groß` zu `größer`, sollte mit dem Word2Vec-Modell die korrekte Antwort `kleiner` berechnet werden können. Die Berechnung ist sehr simpel und erfolgt mit den Wortvektoren.
Sei `X` das gesuchte Wort. Sei *vektor*(`Wort`) der Wortvektor von `Wort`.

Dann ist *vektor*(`X`) = *vektor*(`größer`) - *vektor*(`groß`) + *vektor*(`klein`). Im Allgemeinen entspricht *vektor*(`X`) nicht präzise dem gesuchten Wortvektor. Daher wird das Wort mit der größten Kosinus-Ähnlichkeit zu *vektor*(`X`) ermittelt. In diesem Beispiel ist das *vektor*(`kleiner`).

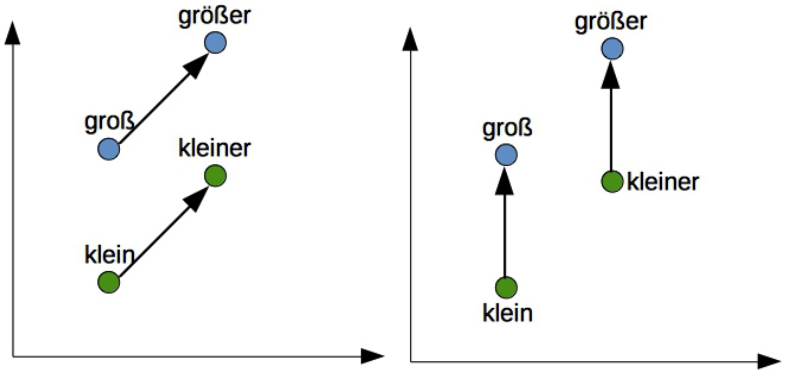

Abbildung 2.2: Darstellung der Komparativ-Relation (links) und der Antonym-Relation (rechts) im zweidimensionalen Raum.
Links: Darstellung der Komparativ-Relation. Die Positive der Wortpaare haben jeweils denselben Abstand zu ihrem Komparativ.
Rechts: Darstellung der Antonymie-Relation. Die Verben der Wortpaare haben jeweils denselben Abstand zu ihrem Gegenwort.

Word2Vec wurde inzwischen in mehreren Gebieten eingesetzt. Folgend sind ein paar Beispiele genannt:

Wolf et al verwenden in [40] die normierten Wortvektoren des CBOW-Modells, um Wörter von einer Sprache in eine andere zu übersetzen. Sie berechnen 2 Modelle unter Verwendung desselben Texts in einer jeweils anderen Sprache. Ein Modell repräsentiert englische Wörter und ein Modell hebräische bzw. arabische Wörter.

Banjade et al verwenden in [41] u.a. Word2Vec-Wortvektoren, um Ähnlichkeiten zwischen Sätzen festzustellen.

Ling et al modifizieren in [42] das CBOW- und Skip-Gram-Modell, um syntaktische Informationen besser erfassen zu können.

Wang & Liao nutzen in [43] CBOW-Wortvektoren zusammen mit Trigrammen und anderen Fehlererkennungs- und Korrekturtechniken. Die Arbeit baut auf einem schon bestehenden Textkorrekturprogramm auf, siehe [44]. Die CBOW-Wortvektoren stellen eine Optimierung dar, und sind nicht Teil der ursprünglichen Architektur.

Mir ist bisher keine Arbeit bekannt, in der einzig oder hauptsächlich Word2Vec zur Fehlererkennung oder Fehlerkorrektur in Texten verwendet wird.

3 Schwellwert-basierte Klassifikatoren

In diesem Kapitel werden mehrere binäre Klassifikatoren vorgestellt. Die Klassifikatoren berechnen für jedes Wort in einem gegebenen Text einen Score. Dieser soll Auskunft darüber geben, ob ein Wort in seinem Wortkontext korrekt oder fehlerhaft ist. In Kapitel 5 wird evaluiert, ob die Klassifikatoren in der Praxis dazu geeignet sind, Fehler in einem Text zu finden.

Schwellwert-basiert heißt, dass ein Wort anhand einer vorgegebenen, numerischen Schranke entweder als *korrekt* oder *fehlerhaft* klassifiziert wird.

Im Folgenden ist die Klassifizierung anhand der berechneten Scores und des Schwellwerts beschrieben.

Klassifikation eines Wortes
Für jedes Wort im Text wurde mit einem Klassifikator ein Score ermittelt. Sei $s \in [0, 1]$ der Schwellwert, ab dem ein Wort als korrekt klassifiziert wird. s ist eine fixe, numerische Schranke für alle Wörter im Text.

Sei w_i das Wort an Stelle i im Text. Sei $s_i \in [0, 1]$ der errechnete Score für w_i.
Ein Wort ist dann korrekt, wenn gilt:

$$s_i > s$$

Ein Wort gilt als fehlerhaft, wenn gilt:

$$s_i \leq s$$

Der erste Abschnitt in diesem Kapitel stellt Klassifikatoren vor, die auf dem Vorgehen des CBOW-Modells basieren. Die Berechnung des Scores erfolgt analog zur Berechnung der Wahrscheinlichkeiten beim Trainieren des neuronalen Netzwerks.
Der zweite Abschnitt stellt Klassifikatoren vor, die auf dem Vektorraum des CBOW-Modells operieren. Der Score wird durch die Betrachtung der Wortvektoren und ihrer Beziehungen zueinander berechnet.

Im Folgenden sei der zu überprüfende Text eine Wortsequenz $(w_1, \ldots, w_D)$, wobei D die Anzahl der Wörter im Text bezeichnet. Ein beliebiges Wort im Text sei mit w_i bezeichnet, wobei i die Stelle im Text angibt, an der das Wort steht, $i \in [1, \ldots, D]$. Der Wortvektor des Wortes w_i sei mit v_i bezeichnet. Der Vektor aus der Kontext-Matrix, welcher zu w_i gehört, sei mit c_i bezeichnet. Sei V die Anzahl der Wörter im Vokabular des CBOW-Modells.
Das Wort, für das der Score berechnet werden soll, sei das x-te Wort im Text. Der

Wortkontext besteht aus den k Wörtern $(w_{x-K}, \ldots, w_{x-1})$ links von w_x, und den k Wörtern $(w_{x+1}, \ldots, w_{x+K})$ rechts von w_x. Sei $l \in [w-K, \ldots w-1, w+1 \ldots, w+K]$. Der Score für w_x wird mit s_x bezeichnet.

3.1 Word2Vec-Wahrscheinlichkeitsbasierte Klassifikatoren

In diesem Abschnitt werden 3 Klassifikatoren vorgestellt. Sie verwenden die Wortvektoren und die Kontextmatrix von CBOW. Damit werden Wort-Kontext-Wahrscheinlichkeiten berechnet, analog zur Berechnung der Wahrscheinlichkeiten beim Trainieren des CBOW-Modells. Im Folgenden werden die berechneten Wort-Kontext-Wahrscheinlichkeiten als Score bezeichnet.

3.1.1 Kontext-basierter Klassifikator

Dieser Klassifikator stellt die Wörter im Kontext von w_x in den Mittelpunkt. Für jedes Wort im Kontext wird ein Score berechnet, dann werden alle Scores zusammengezählt und gemittelt.

Ausgehend vom einzelnen Kontextwort wird der Score $s_{x,l}$ wie folgt berechnet:

$$s_{x,l} = \frac{e^{v_l \cdot c_x}}{\sum_{j=1}^{V} e^{v_l \cdot c_j}} \in [0,1]$$

Der Score s_x für das Wort w_x wird dann wie folgt berechnet:

$$s_x = \frac{1}{2 \cdot k} \cdot \sum_l s_{x,l} \in [0,1]$$

Der so ermittelte Score zeigt an, wie sich das Wort w_x in seinen Kontext einfügt, ausgehend von den Kontextwörtern.

3.1.2 Vorgänger-basierter Klassifikator

Dieser Klassifikator berechnet den Score eines Wortes, indem er nur den Vorgänger des Wortes betrachtet.

Es ist anzunehmen, dass das vorhergehende Wort einen stärkeren Zusammenhang mit dem nachfolgenden Wort hat, als Kontextwörter, die weiter entfernt stehen. Es besteht folglich die Möglichkeit, dass dieser Klassifikator bessere Ergebnisse erzielt als der Kontext-basierte Klassifikator. Andererseits fehlen die Informationen der weiteren Kontextwörter, die beim Trainieren des CBOW-Modells gleichwertig mit dem direkten Nachbarn des Wortes sind. Folglich besteht ebenfalls die Möglichkeit, dass die Ergebnisse dieses Klassifikators schlechter sind als die des Kontext-basierten Klassifikators.

Der Score s_x für das Wort w_x wird wie folgt berechnet:

$$s_x = \frac{e^{v_{x-1} \cdot c_x}}{\sum_{j=1}^{V} e^{v_{x-1} \cdot c_j}} \in [0, 1]$$

Zu beachten ist, dass der Score für das erste Wort im Text durch das nachfolgende Wort berechnet wird.

3.1.3 Mittelvektor Klassifikator

Dieser Klassifikator lehnt sich stark an das Vorgehen beim Trainieren des CBOW-Modells an. Zunächst werden die Wörter im Wortkontext des Wortes, für das der Score ermittelt werden soll, betrachtet. Die Wortvektoren dieser Kontextwörter werden zu einem mittleren Vektor zusammengefasst. Mit diesem mittleren Vektor wird der Score berechnet. Die Berechnung des Scores erfolgt analog zur Berechnung der Wahrscheinlichkeit beim Trainieren des CBOW-Modells.

Das CBOW-Modell wurde so trainiert, dass die berechneten Wahrscheinlichkeiten die Begebenheiten im Trainingskorpus möglichst gut approximieren. Folglich kann man davon ausgehen, dass der Mittelvektor Klassifikator gute Ergebnisse erzielt.

Der mittlere Vektor z_x der Kontextworte wird wie folgt berechnet:

$$z_x = \frac{1}{2 \cdot k} \cdot \sum_{l} v_l$$

Der Score s_x für das Wort w_x wird wie folgt berechnet:

$$s_x = \frac{e^{z_x \cdot c_x}}{\sum_{j=1}^{V} e^{z_x \cdot c_j}} \in [0, 1]$$

3.2 Wortvektorbasierte Klassifikatoren

In diesem Abschnitt werden zwei Klassifikatoren vorgestellt. Zur Ermittlung des Scores werden die Beziehungen zwischen den einzelnen Wortvektoren im Vektorraum betrachtet.

Die Wortvektoren enthalten gewisse Eigenschaften der von ihnen repräsentierten Wörter. Bei den folgenden Klassifikatoren wird davon ausgegangen, dass die Wortvektoren Informationen zur Korrektheit eines Wortes in einem gegebenen Wortkontext enthalten. Diese Information ist in der Beziehung zwischen den Wortvektoren enthalten.

Konkret werden zwei Beziehungen untersucht: die Orientierung der Vektoren und der räumliche Abstand. Diese werden durch die Kosinus-Ähnlichkeit und die Euklidische Distanz ausgedrückt.

3.2.1 Kosinus-Ähnlichkeit Klassifikator

Dieser Klassifikator untersucht die Ähnlichkeit der Wörter, indem er die Orientierung der Wortvektoren betrachtet. Als Maß für die Orientierung wird die Kosinus-Ähnlichkeit verwendet. Eine Kosinus-Ähnlichkeit von 1 zwischen zwei Vektoren bedeutet, dass die Vektoren dieselbe Orientierung haben. Eine Kosinus-Ähnlichkeit von -1 bedeutet, dass zwei Vektoren genau gegensätzlich orientiert sind. Es wird davon ausgegangen, dass eine gleiche oder ähnliche Orientierung der Wortvektoren auf einen Zusammenhang im Satzkontext hinweist. Das heißt, die Wörter sind im untersuchten Wortkontext korrekt.

Mikolov et al wenden dieses Vorgehen in [38] in einem thematisch anderen Kontext erfolgreich an, siehe auch Abschnitt 2.4.

Der Score s_x für das Wort w_x wird wie folgt berechnet:

Zunächst wird der mittlere Vektor z_x der Kontextwörter berechnet:

$$z_x = \frac{1}{2 \cdot k} \sum_l v_l$$

Die Kosinus-Ähnlichkeit d_x zwischen z_x und v_x wird wie folgt berechnet:

$$d_x = \frac{z_x \cdot v_x}{\|z_x\| \cdot \|v_x\|} \in [-1, 1]$$

Zur Berechnung des Scores s_x wird die Kosinus-Distanz auf das Intervall $[0, 1]$ projiziert:

$$s_x = 0,5 \cdot d_x + 0,5 \in [0, 1]$$

Diese Projektion findet aus Konsistenzgründen statt: Liegen alle berechneten Scores in dieser Arbeit im Intervall [0,1], können die Ergebnisse leichter verglichen werden.

3.2.2 Euklidische Distanz Klassifikator

Der Klassifikator im vorangehenden Abschnitt hat die Orientierung der Wortvektoren betrachtet. Der Klassifikator in diesem Abschnitt betrachtet den räumlichen Zusammenhang zwischen den Wortvektoren. Als Maß für den Abstand zweier Vektoren wird die euklidische Distanz verwendet. Im Folgenden wird davon ausgegangen, dass eine niedrige euklidische Distanz auf einen Zusammenhang im Satzkontext hinweist. Das heißt, die Wörter sind im betrachteten Wortkontext korrekt. Dies entspricht der räumlichen Vorstellung, dass solche Wortvektoren nahe beieinander liegen.

Der Score s_x für das Wort w_x im Text wird wie folgt berechnet:

Zunächst wird der mittlere Vektor z_x der Kontextwörter berechnet:

$$z_x = \frac{1}{2 \cdot k} \sum_l v_l$$

Im nächsten Schritt wird die euklidische Distanz d_x zwischen dem mittleren Vektor z_x und dem Wortvektor v_x berechnet:

$$d_x = \|z_x - v_x\|_2 \in [0, \infty)$$

Aus Konsistenzgründen wird d_x auf das Intervall $[0, 1]$ projiziert. Dazu wird eine maximal mögliche euklidische Distanz d_{max} angenommen. Sei n die Dimension des Vektorraums. d_{max} ist wie folgt definiert:

$$d_{max} = n$$

d_{max} wurde durch Experimente mit den für diese Arbeit berechneten CBOW-Modellen ermittelt.

Die Projektion p_x der berechneten euklidischen Distanz d_x wird wie folgt berechnet:

$$p_x = \frac{d_x}{d_{max}} \in [0, 1]$$

Der Score s_x für das Wort w_x wird dann wie folgt berechnet:

$$s_x = 1 - p_x \in [0, 1]$$

Für s_x gilt, dass ein hoher numerischer Wert eine niedrige euklidische Distanz impliziert. Somit ist die Bedeutung der Werte im Intervall $[0, 1]$ gleich wie bei den anderen vorgestellten Klassifiaktoren. Je höher s_x ist, desto wahrscheinlicher ist w_x im gegebenen Kontext korrekt.

4 Experimente

Alle Experimente wurden unter identischen Rahmenbedingungen durchgeführt. Das betrifft die Hardware, verwendete Software, die Validierungsdaten sowie die Implementierung und Ausführung der Algorithmen.

Die Experimente sollen klären, wie sich die in Kapitel 3 vorgestellten Klassifikatoren mit verschiedenen Schwellwerten verhalten. Außerdem soll geklärt werden, ob die Verwendung eines domänenspezifischen Textkorpus zum Trainieren von CBOW andere Ergebnisse liefert als die Verwendung eines domänenunabhängigen Textkorpus. Anhand der Ergebnisse wird ebenfalls ersichtlich sein, ob das Word2Vec-Modell und die Word2Vec-Wortvektoren zusammen mit einer numerischen Klassifiaktions-Schranke zur Text-Fehlererkennung geeignet sind.

Zusätzlich zu den beiden Parametern Klassifikations-Schranke und Wahl des Modells wurden noch weitere Parameter eingeführt. Dazu zählen beispielsweise das Ignorieren von Punktuationszeichen im Wortkontext oder die Möglichkeit, die Größe des Wortkontexts zu variieren. Diese Parameter sollen einerseits dazu beitragen, nicht direkt relevante Informationen aus der Evaluierung rauszunehmen, wie Punktuationszeichen und Zahlen. Andererseits soll der Klassifikator so optimal wie möglich arbeiten, was unter anderem durch die Betrachtung verschiedener Wortkontexte erreicht werden kann.

4.1 Versuchsaufbau

In diesem Abschnitt werden die Randdaten für die Evaluierung der in Kapitel 3 vorgestellten Klassifikatoren vorgestellt. Diese reichen von den technischen Rahmenbedingungen bis zu den verwendeten Texten.

Technische Daten

In diesem Unterabschnitt geht es um die technischen Rahmenbedingungen der Experimente. Dazu zählen der zur Berechnung verwendete Computer sowie verwendete Programme und Skripte.

Verwendeter Computer:
MacBook Pro, Baujahr 2012 (Mitte des Jahres)
Prozessor: 2,9 GHz Intel Core i7
RAM: 8GB
Betriebssystem: OS X Yosemite (Version 10.10.5)

Verwendete Programmiersprachen, Programme und Skripte:
Programmiersprache: Python 3.4
Entwicklungsumgebung: Eclipse (Version 4.2.1)[5] mit Python-Erweiterung PyDev[6]
Verwendete word2vec-Implementierung: Gensim[7]
Skript zur Extraktion von Wikipedia-xml-Dateien in txt-Dateien: WikipediaExtractor 2.32 von Giuseppe Attardi[8]
Weitere verwendete Pakete, welche nicht von vornherein in python enthalten sind: numpy[9]

Textkorpora

Das CBOW-Modell wurde mit zwei unterschiedlichen Textkorpora trainiert. Dadurch kann der Einfluss eines domänenspezifischen und eines domänenunabhängigen Textkorpus auf die Genauigkeit eines Klassifikators beurteilt werden. Die Modelle wurden mit Gensim erstellt und trainiert. Gensim ist eine Bibliothek für python, welche Methoden für Topic Modelling mit großen Textkorpora bereitstellt. Im Folgenden sind die beiden Textkorpora näher beschrieben.

Wikipedia-Korpus

Zum Trainieren des Wikipedia-Modells wurden die Artikel der deutschen Wikipedia[10] verwendet. Die xml-Dateien, welche online zur Verfügung stehen[11], wurden mit dem Skript *WikipediaExtractor* in Fließtext-Form, ohne xml-Tags, gebracht. *WikipediaExtractor* entfernt zusätzlich zu den xml-Tags auch andere Annotationen und Elemente, wie Tabellen, Bilder etc. Was übrig bleibt, ist ein Text ohne Annotationen, mit der Ausnahme eines öffnenden und schließenden HTML-Tags für jeden Artikel.

Zeitungsartikel-Korpus

Für das Zeitungsartikel-Modell wurden digitalisierte Zeitungsartikel aus dem Jahr 1914 verwendet. Diese wurden im Zuge des Projekts *Ostbayern im I. Weltkrieg. Die „Donau-Zeitung“ digital*[12] digitalisiert.

Alle Trainingstexte wurden manuell auf Rechtschreibfehler und semantische Fehler überprüft und ggf. korrigiert. Das Modell wurden nur mit Artikeln trainiert, die nicht zur Validierung der Klassifikatoren verwendet wurden. Es wurden 583 korrigierte Artikel, bestehend aus insgesamt 121 306 Textelementen wie Wörtern und Punktuationszeichen, verwendet.

[5] `http://www.eclipse.org/` – zuletzt abgerufen am: 24. 09. 2015
[6] `http://www.pydev.org/` – zuletzt abgerufen am: 24. 09. 2015
[7] `https://radimrehurek.com/gensim/index.html` – zuletzt abgerufen am: 20. 12. 2015
[8] `http://medialab.di.unipi.it/wiki/Wikipedia_Extractor` – zuletzt abgerufen am: 24. 09. 2015
[9] `http://www.numpy.org/` – zuletzt abgerufen am: 24. 09. 2015
[10] Download vom 30. 06. 2015
[11] Wikipedia-Dumps sind online verfügbar unter `https://dumps.wikimedia.org/dewiki/latest/`
[12] `http://dz1914.uni-passau.de/` – zuletzt abgerufen am: 24. 09. 2015 und `http://www.phil.uni-passau.de/die-fakultaet/lehrstuehle-professuren/rehbein/forschung/donau-zeitung-digital.html` – zuletzt abgerufen am 24. 06. 2015

Trainingseinstellungen der Modelle

Die Parameter beim Zeitungsartikel-Modell sind deutlich niedriger als beim Wikipedia-Modell. Dies ist durch den deutlich kleineren Textkorpus begründet.
Da weniger Trainingsdaten vorhanden sind, stehen auch weniger Daten zum Erlernen der Vektorparameter zur Verfügung. Es liegt nahe, dass weniger Eigenschaften erfasst werden können. Daher macht es Sinn, die Dimension der Wortvektoren niedriger zu wählen als beim Wikipedia-Modell.
Das minimale Vorkommen eines Wortes, um ins Vokabular des Zeitungsartikel-Modells aufgenommen zu werden, ist ebenfalls niedriger. Abgesehen von der deutlich kleineren Textmenge ist dieses Vorgehen dadurch begründet, dass die Texte manuell korrigiert wurden. Die Anzahl der fehlerhaften Wörter im Trainingskorpus dürfte gering sein.

In der folgenden Tabelle sind die voneinander abweichenden Trainingsparameter der Modelle angegeben.

	Wikipedia	**Zeitungsartikel**
Dimension der Wortvektoren	100	20
Minimales Vorkommen eines Wortes, um im Vokabular des Modells enthalten zu sein	20	5
Wörter im Trainingskorpus	690 518 796	121 306
Wörter im Vokabular des Modells	649 461	2 732

Validierungsdaten

Als Validierungstext wurden Zeitungsartikel aus der Donau-Zeitung von Anfang des 20. Jahrhunderts verwendet, siehe auch unter 4.1 – *Textkorpora – Zeitungsartikel-Modell*. Diese Zeitungartikel, welche in Frakturschrift gedruckt worden waren, wurden mit dem OCR-Programm ABBY[13] digitalisiert.

Die Digitalisierung der Texte war nicht fehlerfrei. Die fehlerhaften Wörter lassen sich in 4 Klassen einteilen:

1. Einige Wörter stellen existierende Wörter dar, die jedoch nicht korrekt sind. Beispielsweise *ungefragt*, korrekt wäre *angefragt*. *setzt*, korrekt wäre *jetzt*. *er*, korrekt wäre *der*. Oder *Ratte* statt korrekterweise *Flotte*. In manchen Fällen handelt es sich um eine falsch digitalisierte Groß- oder Kleinschreibung. Insgesamt sind ca. 12 % der fehlerhaften Wörter existierende Wörter, die keinen Bindestrich enthalten.

2. Einige Wörter wurden so digitalisiert, dass sie keinen Sinn mehr ergeben. Beispielsweise *Reibungne*, korrekt wäre *Reibungen*. *Auust*, korrekt wäre *August*. *sWrafen*, korrekt wäre *aufgerufen*. Oder *Paffe*, korrekt wäre *Pässe*. Letzterer Fall dürfte durch das Fraktur-s zustande gekommen sein, das einem f sehr ähnlich sieht. Diese Wortklasse enthält 14-15% der fehlerhaften Wörter.

3. Viele Wörter sind deshalb fehlerhaft, weil sie mit einem Bindestrich im Wort digitalisiert wurden.

[13] `http://www.abbyy.de/` – zuletzt abgerufen am 24.06.2015

So beispielsweise *Ent-scheidung*, korrekt wäre *Entscheidung*. Oder *militäri-schen*, korrekt wäre *militärischen*. Diese Wörter stellen 68,7% der fehlerhaften Wörter dar. Die meisten Bindestriche dürften nicht auf eine fehlerhafte Digitalisierung zurückgehen. Sie stammen von Zeilen- bzw. Wortumbrüchen im gedruckten Text, welche per Bindestrich getrennt wurden. In einem digitalen Text sind solche Wörter jedoch fehlerhaft.

4. In einigen Fällen wurden Buchstaben als Sonderzeichen digitalisiert, beispielsweise *geplante»* statt korrekterweise *geplanten*. In manchen Fällen wurden Punktuationszeichen falsch digitalisiert, beispielsweise *,* statt *..* In fünf Fällen wurden Zahlen falsch digitalisiert. Statt einer Zahl steht dann ein einzelner Buchstabe im Text, wie beispielsweise *L* statt einer *2*. In drei Fällen steht ein einzelner Buchstabe als fehlerhaft im Text, beispielsweise *W* statt *am*. Diese Fehlerklasse enthält ca. 5% der fehlerhaften Wörter.

Die Fehlerhaftigkeit dieser Wörter müsste aus dem Kontext, in dem das Wort steht, ersichtlich sein. Vor allem bei Wörtern aus der ersten Klasse ist eine Fehlerkorrektur auf Kontextbasis interessant. Bei den anderen drei Klassen dürften viele Wörter durch den Vergleich mit einem Wörterbuch als fehlerhaft erkannt werden.

Folgend sind zwei Sätze aus einem Zeitungsartikel vom 07. 08. 1914 abgedruckt. Fehlerhafte Wörter sind hervorgehoben. Die Sätze enthalten fehlerhafte Wörter mit Bindestrich und solche, die keinen Sinn mehr ergeben.

```
Bekanntmachung üSer den Uebergang der vollziehenden Gewalt auf die
Militärbefehlshaber.
[...]
Mr Verkehr mit Verkehrsmitteln jeder Art über die Auslandsgrenze wird
nur über fol-gende Ueberwachungsstellen zugelassen:
[...]
```

Zur Validierung der in Kapitel 3 vorgestellten Klassifikatoren wurden 203 fehlerhafte Zeitungsartikel zufällig ausgewählt. Zu jedem Zeitungsartikel existiert eine Version, welche manuell hinsichtlich syntaktischer und semantischer Fehler korrigiert wurde. Außerdem wurden alle Bindestriche entfernt, welche durch Wortumbrüche entstanden sind. Diese korrigierten Texte werden bei der Validierung zur Überprüfung der Klassifikatoren verwendet.
Die fehlerhaften Texte bestehen aus insgesamt 68 840 Textelementen, davon sind 9 648 Punktuationszeichen und 935 Zahlen, der Rest sind Wörter. Im Durchschnitt besteht ein Zeitungsartikel aus 339 Textelementen. Davon sind 287 Wörter, wovon 4 Wörter (1,4%) fehlerhaft sind. Durchschnittlich sind pro Zeitungsartikel 47-48 Punktuationszeichen und 4-5 Zahlen enthalten.

Wie man hier sieht, ist der Datensatz sehr unbalanciert. Die meisten Wörter sind korrekt, und vergleichsweise sehr wenige Wörter (1,4% aller Wörter) beinhalten einen Schreib- oder Digitalisierungsfehler.

4.2 Vorverarbeitung

Bevor mit den Traningsdaten CBOW-Modelle trainiert werden, werden die Trainingstexte bearbeitet. Dasselbe gilt für die Validierungstexte, bevor diese zur Evaluierung der Klassifikatoren benutzt werden.

Hauptziel der Vorverarbeitung ist, alle Wörter und Satzzeichen durch ein Leerzeichen voneinander zu trennen. Eine Ausnahme stellen Bindestriche dar, welche Wörter wie beispielsweise `Österreich-Ungarn` verbinden. Jedes so getrennte Satzelement ist später ein Element, mit dem das Modell trainiert wird bzw. das zur Validierung eines Klassifikators verwendet wird. Dadurch wird vermieden, dass eigentlich gleiche Wörter verschiedene Wortklassen bilden, die sich nur durch angehängte Satzzeichen unterscheiden.
Beispielsweise das Wort `Stadt`: steht dieses Wort am Ende eines Satzes, würden ohne Trennung die Wortklassen `Stadt.`, `Stadt!` und `Stadt?` entstehen. Dadurch würden beim Trainieren des CBOW-Modells semantische Informationen für die Klasse `Stadt` verloren gehen. Außerdem kann es sein, dass `Stadt!` so selten am Satzende steht, dass es nicht ins Vokabular aufgenommen wird. Wenn `Stadt!` dann in einem Validierungstext vorkommt, kann es nicht bearbeitet werden, da das Modell das Wort so nicht kennt.

Zahlen werden durch einen Platzhalter (hier: `__NUM__`) ersetzt. Tabs werden durch Leerzeichen ersetzt und mehrfache Leerzeichen durch ein einziges.
Beim Trainingskorpus für das Wikipedia-Modell werden alle noch bestehenden HTML-Tags, `< ... >` und `<\... >`, gelöscht.
Bei den Validierungstexten werden alle Zeilenumbrüche entfernt.

Folgend ist ein Ausschnitt aus einem Text aus der Wikipedia abgedruckt. Der Text wurde bereits durch das Skript *WikipediaExtractor* bearbeitet.

Zunächst die nicht vorverarbeitete Version:

```
<doc id=„16079" url=„"http://de.wikipedia.org/wiki?curid=16079" title
=„Dateiattribut">

Dateiattribut

Unter Dateiattributen (oder auch Dateieigenschaften) versteht man [...].
Einige Betriebssysteme (wie Mac OS) haben ein wesentlich
umfangreicheres Metadatenkonzept, und unterstützen so z. B. auch die
Speicherung Dateityp-spezifischer Informationen [...] Daten dieser Art direkt
in den Inhalten der Dateien unterzubringen (z. B. ID3-Tags in MP3-Dateien).

</doc>
```

Folgend der vorverarbeitete Text:

```
Dateiattribut
Unter Dateiattributen ( oder auch Dateieigenschaften ) versteht man [...]
. Einige Betriebssysteme ( wie Mac OS ) haben ein wesentlich
umfangreicheres Metadatenkonzept , und unterstützen so z .  B . auch die
Speicherung Dateityp-spezifischer Informationen [...] Daten dieser Art
direkt in den Inhalten der Dateien unterzubringen ( z .  B .
ID__NUM__-Tags in MP__NUM__-Dateien ) .
```

4.3 Allgemeine Eingabe-Parameter

Dieser Abschnitt beschreibt Parameter, welche für die Berechnung des Scores verwendet werden. Diese Parameter werden, mit einer Ausnahme, bei der Berechnung der Ergebnisse von jedem Klassifikator verwendet.

WORTFENSTER Natürliche Zahl größer gleich 1. Das Wortfenster ist symmetrisch und gibt an, wie viele Wörter links und rechts neben dem aktuellen Wort zur Berechnung des Scores verwendet werden. Ist das Wortfenster beispielsweise 3, so werden die 3 Wörter links und die 3 Wörter rechts von dem Wort, welches aktuell betrachtet wird, verwendet.

EINZELNER_SATZ Parameter, der angibt, ob ein Punkt, Fragezeichen oder Ausrufezeichen das Wortfenster beendet. Diesem Parameter liegt zugrunde, dass zwei hintereinander stehende Sätze nicht unbedingt etwas miteinander zu tun haben müssen. D.h., die Betrachtung des Wortkontextes endet mit dem Satzende. Kein Wort wird über seinen Satzkontext hinaus betrachtet.

KEINE_ZAHLEN Gibt an, ob Zahlen während der Berechnung des Scores beachtet werden. Falls nicht, wird keiner Zahl ein Score zugeordnet. Ebenso fließt während der Scoreberechnung eines Wortes keine Zahl in die Berechnung mit ein.

KEINE_PUNKTUATION Gibt an, ob Punktuationszeichen während der Scoreberechnung beachtet werden. Falls nicht, wird keinem Satzzeichen ein Score zugeordnet. Ebenso fließt während der Scoreberechnung eines Wortes kein Satzzeichen in die Berechnung mit ein. Einzige Ausnahme ist der oben genannte Parameter *EINZELNER_SATZ*. Ist dieser Parameter *True*, so endet das Wortfenster entsprechend der Parameterbeschreibung. Die Satzzeichen fließen aber nicht in die Berechnung mit ein.

Folgende Satzzeichen werden hier betrachtet:
„"‚‘€ % ‰ ! "# $ & ' ( ) * + , . / : ; > < = ? @ [ ] \^ { } | ~

NUR_VOKABULARWÖRTER Wenn diese Option aktiviert (*True*) ist, werden Wörter, welche nicht im Vokabular des Modells vorhanden sind, vor der Berechnung des Scores aus dem Text entfernt.

Ist diese Option nicht aktiviert (*False*), dann werden alle Wörter, welche nicht im Vokabular enthalten sind, automatisch als falsch bzw. fehlerhaft klassifiziert. Auch können Wörter ausserhalb des Vokabulars im Wortfenster eines Wortes stehen. Somit beeinflussen sie indirekt die Scoreberechnung des Wortes, da ein Wert „weniger“ zur Berechnung vorhanden ist.

4.4 Ausgabe-Kennzahlen

Zur Bewertung der Klassifikatoren wird das F1-Maß verwendet.
Die positive Klasse ist als die Menge der Wörter definiert, welche fehlerhaft sind. Das schließt alle Digitalisierungsfehler mit ein. Ebenfalls sind in der positiven Klasse alle Wörter enthalten, die korrekterweise mit einem Bindestrich digitalisiert wurden, wenn dieser Bindestrich zu einem fehlerhaften Wort führt.
Demgegenüber beinhaltet die negative Klasse alle korrekten Wörter.

Entsprechend sind die Klassen definiert, bei denen ein Wort richtig klassifiziert wird:

True-Positive Wörter mit Digitalisierungsfehler, oder nicht korrektem Bindestrich, welche als solche klassifiziert werden. In den Tabellen mit *True-Pos.* abgekürzt.

True-Negative Wörter ohne Digitalisierungs- oder Schreibfehler, welche als korrekt klassifiziert werden. In den Tabellen mit *True-Neg.* abgekürzt.

Die Klassen, bei denen ein Wort nicht richtig klassifiziert wird, sind wie folgt definiert:

False-Negative Wörter mit Digitalisierungsfehler oder nicht korrektem Bindestrich, welche als korrekt klassifiziert werden. In den Tabellen mit *False-Neg.* abgekürzt.

False-Positive Korrekte Wörter, welche als Wörter mit Digitalisierungsfehler klassifiziert werden. In den Tabellen mit *False-Pos.* abgekürzt.

Aus diesen Klassen leiten sich zwei Kennzahlen ab. Mit diesen können grundlegende Aussagen über die Genauigkeit und die Trefferquote des Klassifikators getroffen werden:

Precision Der Anteil der Wörter in der positiven Klasse, welche tatsächlich fehlerhaft sind. Die Kennzahl wird wie folgt berechnet:

$$\frac{\text{True-Positive}}{\text{True-Positive} + \text{False-Positive}} \in [0, 1]$$

Recall Der Anteil der Wörter mit Digitalisierungsfehler oder falschem Bindestrich, welche als solche klassifiziert werden. Die Kennzahl wird wie folgt berechnet:

$$\frac{\text{True-Positive}}{\text{True-Positive} + \text{False-Negative}} \in [0, 1]$$

Der **F1-Score** wird dann wie folgt berechnet:

$$2 \cdot \frac{precision \cdot recall}{precision + recall} \in [0, 1]$$

Bei der Erhebung der Daten werden die Evaluierungstexte als ein zusammenhängender Text gesehen. Der F1-Score wird also gesamt für alle Texte berechnet, nicht für jeden Text einzeln.

5 Evaluierung

Im ersten Abschnitt werden ausgewählte Ergebnisse der durchgeführten Experimente dargestellt. Zusätzlich werden diejenigen Wörter beschrieben, die nicht im Vokabular CBOW-Modells enthalten sind.

Im zweiten Abschnitt werden die Ergebnisse diskutiert. Am Beginn des Abschnitts befindet sich ein eigener Unterabschnitt über die Wörter, welche nicht im Vokabular eines Modells enthalten sind.

5.1 Ergebnisse

In diesem Abschnitt werden die Ergebnisse der durchgeführten Experimente vorgestellt. Mit jedem Klassifikator wurde eine Vielzahl an Testläufen durchgeführt. Bei jedem Testlauf wurden die Eingabe-Parameter variiert. Im Folgenden ist eine Auswahl der Ergebnisse zu jedem Klassifikator abgebildet.

Der Abschnitt ist nach Klassifikatoren gegliedert. In jedem Unterabschnitt befinden sich Ergebnisse zu jeweils einer Klassifikatoren-Gruppe.

Parameter-Standard-Einstellung

Durch gleiche Parametereinstellungen sind die Ergebnisse besser vergleichbar. Im folgenden sind die Parametereinstellungen, sofern beim konkreten Ergebnis nicht anders beschrieben, wie folgt:
`WORTFENSTER` = 5
`EINZELNER_SATZ` = Falsch
`KEINE_ZAHLEN` = Wahr
`KEINE_PUNKTUATION` = Wahr
`NUR_VOKABULARWÖRTER` = Wahr
Diese Einstellungen werden im folgenden als *Parameter-Standard-Einstellung* bezeichnet.

Eine Änderung dieser Parametereinstellungen hat keine grundlegende Änderung der Resultate bewirkt. Einzige Ausnahme stellt der Parameter `NUR_VOKABULARWÖRTER` dar. Ist dieser Wahr, so wird ein großer Teil der Wörter aus der positiven Klasse nicht beachtet. Daher ist im Folgenden pro Modell ein Ergebnis mit `NUR_VOKABULARWÖRTER` = Falsch angegeben.

Wörter, welche nicht im Vokabular enthalten sind

Hier werden die Wörter analysiert, welche nicht im Vokabular des jeweiligen Modells enthalten sind. Bei der Darstellung der Ergebnisse ist pro Modell ein Ergebnis

abgebildet, das auch diese Wörter enthält. Dies entspricht der Parametereinstellung `NUR_VOKABULARWÖRTER` = Falsch. Bei allen anderen Ergebnissen wurden diese Wörter vor der Berechnung entfernt, da sie für jeden Klassifikator dieselben sind und immer gleich klassifiziert wird.

Wikipedia-Modell

In Tabelle 5.1 auf Seite 31 ist der Anteil der Wörter in der positiven und negativen Klasse angegeben, der nicht im Vokabular des Wikipedia-Modells enthalten ist. In der positiven Klasse der Schreib- bzw. Digitalisierungsfehler sind 85,5% der Wörter nicht im Vokabular enthalten. Das sind insgesamt 709 Wörter bzw. 680 voneinander verschiedene Wörter. In der negativen Klasse der korrekten Wörter sind 3,3% der Wörter nicht im Vokabular enthalten. Das entspricht insgesamt 2 266 Wörtern oder 1 939 voneinander verschiedenen Wörtern.
Siehe Abschnitt 4.1 – *Validierungsdaten* für die Zusammensetzung der positiven Klasse. Die Wörter aus der negativen Klasse, welche nicht im Vokabular des Wikipedia-Modells enthalten sind, setzen sich aus verschiedenen Worttypen zusammen. Viele Wörter gehören zu den folgenden Worttypen:

1. Eigennamen, welche heute teils nicht mehr gebräuchlich sind. Zum Beispiel *Philippreuth*, *Kirnthaler*.
2. Bezeichnungen, welche heute nicht mehr oder selten gebräuchlich sind. Zum Beispiel *Häuslerssohn*.
3. Begriffe für Einrichtungen oder Techniken. Zum Beispiel *Brieftaubenverkehr*, *Telegraphenlinien*, *Reichsratskammer*, *Dreiverband*.
4. Zusammengesetzte Wörter. Zum Beispiel *Brieftauben-Liebhaber-Vereine*.
5. Schreibweisen oder Begrifflichkeiten, welche heute nicht mehr oft verwendet werden. Zum Beispiel *Totalmobilisierung*, *unterfertigten*.
6. Wörter mit ß statt dem heute üblichen ss.
7. Wörter, welche statt ä, ö, ü ein ae, oe, ue enthalten. Die Schreibweise der Umlaute ist in den zugrunde liegenden, digitalen Zeitungsartikeln nicht konsequent durchgeführt worden. Ein Beispiel hierfür ist *Oesterreich-Ungarn* bzw. *Österreich-Ungarn*

Ein weiteres Problem stellen Wörter dar, welche z.B. in verschiedenen Verbform, in verschiedenen Zeitformen oder in Plural bzw. Einzahl geschrieben wurden. Beispielsweise werden die Wörter *anstandslose*, *Sammelorten*, *Größenwahns* oder *pünktlichste* nicht erkannt.

Zeitungsartikel-Modell

In Tabelle 5.2 auf Seite 33 ist der Anteil der Wörter in der positiven und negativen Klasse angegeben, der nicht im Vokabular des Zeitungsartikel-Modells enthalten ist. In der positiven Klasse der Schreib- bzw. Digitalisierungsfehler sind 94,69% der Wörter nicht im Vokabular enthalten. Das sind 785 von 829 Wörtern. Die 785 Wörter bestehen aus 745 voneinander verschiedenen Wörtern. In der negativen Klasse der korrekten Wörter sind 22,4% der Wörter nicht im Vokabular enthalten. Das sind 15 266 Wörter, bestehend aus 10 448 voneinander verschiedenen Wörtern.
Siehe Abschnitt 4.1 – *Validierungsdaten* für die Zusammensetzung der positiven Klasse.

Die Wörter aus der negativen Klasse, welche nicht im Vokabular des Zeitungsartikel-Modells enthalten sind, setzen sich nicht aus einigermaßen klar umrissenen Worttypen zusammen, wie beim Wikipedia-Modell. Zu diesen Wörtern zählen Institutionen und Substantive wie beispielsweise *Militärbehörde*, *Gemeindebehörden*, *Staatsdienst*, *Tiere*, *Gemüse* und *Privatverkehr*. Es ist zu beobachten, dass der Plural häufiger nicht im Vokabular enthalten ist als der Singular. Adjektive wie beispielsweise *treue*, *hergestellten* und *enttäuschen* sind ebenfalls nicht im Vokabular enthalten. Dazu kommen noch Verben in verschiedenen Formen, beispielsweise im Infinitiv. Viele Lebensmittel und Tierarten sind ebenfalls nicht im Vokabular enthalten, wie *Butter*, *Rahm*, *Sellerie*, *Federvieh*, *Rindvieh* und *Schweine*.
Im Gegensatz zum Wikipedia-Modell werden Wörter mit ß vom Modell erkannt.

5.1.1 Word2Vec-Wahrscheinlichkeitsbasierte Klassifikatoren

Kontext-basierter Klassifikator

Ergebnis 1: Kontext-basierter Klassifikator, Wikipedia-Modell

Die Parametereinstellungen weichen von der Standard-Parameter-Einstellung (siehe Anfang Abschnitt 5.1) ab. Dadurch werden auch diejenigen Wörter bei der Berechnung des Scores berücksichtigt, welche nicht im Vokabular des Wikipedia-Modells enthalten sind. Die Parameter sind wie folgt gesetzt:
`WORTFENSTER` = 5
`EINZELNER_SATZ` = Falsch
`KEINE_ZAHLEN` = Falsch
`KEINE_PUNKTUATION` = Falsch
`NUR_VOKABULARWÖRTER` = Falsch

In Abbildung 5.1 und Tabelle 5.1, siehe rechts, sind die Daten zu Ergebnis 1 dargestellt. Das Experiment wurde für mehrere Schwellwerte zwischen 10^{-5} bis 10^{-16} wiederholt.

Je kleiner der Schwellwert wird, desto niedriger ist der Recall und die Anzahl der True-Positives. Gleichzeitig steigt die Anzahl der True-Negatives an, je kleiner der Schwellwert wird.

Der Recall ist durch die Wörter, die nicht im Vokabular enthalten sind, nach unten beschränkt. Daher kann er nicht unter 85,5% sinken. Je kleiner der Schwellwert wird, desto mehr nähert sich der Recall der unteren Schranke an, bis er sie bei 10^{-15} vollständig erreicht hat und ab 10^{-12} nur um ein Wort abweicht.

Obwohl der Recall seine Unterschranke (fast) erreicht hat, steigt die Precision mit Verringerung des Schwellwerts weiter an. Es werden also nicht mehr fehlerhafte Wörter gefunden, aber mehr korrekte Wörter als korrekt klassifiziert. Die maximal erreichte Precision beträgt 0,2383.

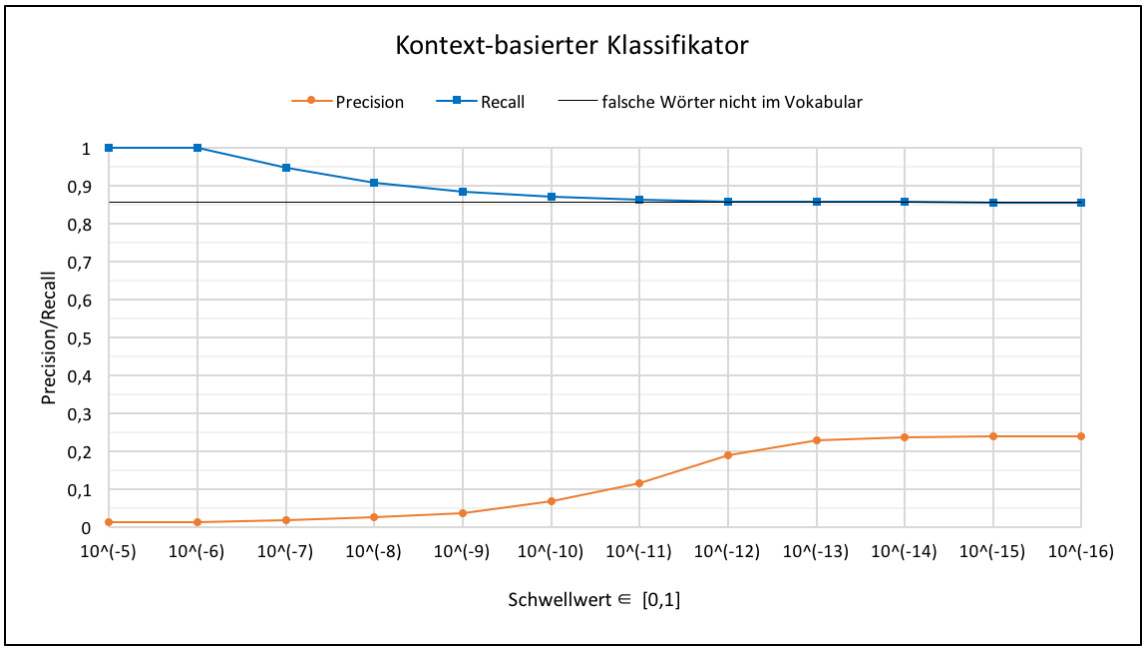

Abbildung 5.1: Diagramm mit Recall und Precision, Ergebnis 1, Kontext-basierter Klassifikator, Wikipedia-Modell. Auch Wörter, die nicht im Vokabular enthalten sind, wurden bei der Berechnung berücksichtigt.

Tabelle 5.1: Tabelle mit weiteren Ausgabe-Kennzahlen, Ergebnis 1, Kontext-basierter Klassifikator, Wikipedia-Modell. Auch Wörter, die nicht im Vokabular enthalten sind, wurden bei der Berechnung berücksichtigt.

Schwellwert $\in [\mathbf{0}, \mathbf{1}]$	True-Neg.	True-Pos.	davon nicht im Vokabular	False-Pos.	davon nicht im Vokabular	False-Neg.
$\mathbf{10^{-5}}$	50	829	709 (85,52%)	67961	2266 (3,33%)	0
$\mathbf{10^{-6}}$	339	829	709 (85,52%)	67672	2266 (3,35%)	0
$\mathbf{10^{-7}}$	24099	784	709 (90,43%)	43912	2266 (5,16%)	45
$\mathbf{10^{-8}}$	38940	753	709 (94,16%)	29071	2266 (7,79%)	76
$\mathbf{10^{-9}}$	49008	732	709 (96,86%)	19003	2266 (11,92%)	97
$\mathbf{10^{-10}}$	58236	721	709 (98,34%)	9775	2266 (23,18%)	108
$\mathbf{10^{-11}}$	62583	714	709 (99,30%)	5428	2266 (41,75%)	115
$\mathbf{10^{-12}}$	64964	710	709 (99,86%)	3047	2266 (74,37%)	119
$\mathbf{10^{-13}}$	65609	710	709 (99,86%)	2402	2266 (94,34%)	119
$\mathbf{10^{-14}}$	65723	710	709 (99,86%)	2288	2266 (99,04%)	119
$\mathbf{10^{-15}}$	65744	709	709 (100%)	2267	2266 (99,96%)	120
$\mathbf{10^{-16}}$	65745	709	709 (100%)	2266	2266 (100%)	120

Ergebnis 2: Kontext-basierter Klassifikator, Zeitungsartikel-Modell

Die Parametereinstellungen weichen von der Standard-Parameter-Einstellung (siehe Anfang Abschnitt 5.1) ab. Dadurch werden auch diejenigen Wörter bei der Berechnung des Scores berücksichtigt, welche nicht im Vokabular des Zeitungsartikel-Modells enthalten sind. Die Parameter sind wie folgt gesetzt:
`WORTFENSTER` = 5
`EINZELNER_SATZ` = Falsch
`KEINE_ZAHLEN` = Falsch
`KEINE_PUNKTUATION` = Falsch
`NUR_VOKABULARWÖRTER` = Falsch

In Abbildung 5.2 und Tabelle 5.2, siehe rechts, sind die Daten zu Ergebnis 2 dargestellt. Das Experiment wurde für mehrere Schwellwerte zwischen $0,0004$ bis $0,0001$ wiederholt. Die Schwellwerte, bei denen die Wörter unterschiedlich klassifiziert werden, umfassen einen kleineren Wertebereich als die Schwellwerte beim Wikipedia-Modell.

Je kleiner der Schwellwert wird, desto niedriger ist der Recall und die Anzahl der True-Positives. Gleichzeitig steigt die Anzahl der True-Negatives an, je kleiner der Schwellwert wird.

Gleich wie bei Ergebnis 1 nähert sich der Recall immer weiter seiner unteren Schranke, hier 94,69%, bis er sie bei 0,00022 erreicht hat. Die Precision steigt, wie bei Ergebnis 1, auch bei einer kleiner werdenden Schranke noch an. Die Steigung ist jedoch sehr gering und die Precision bleibt immer unter 0,05.

Die Anzahl der True-Negatives steigt weiter sichtbar an, auch wenn der Recall seine untere Schranke erreicht hat.

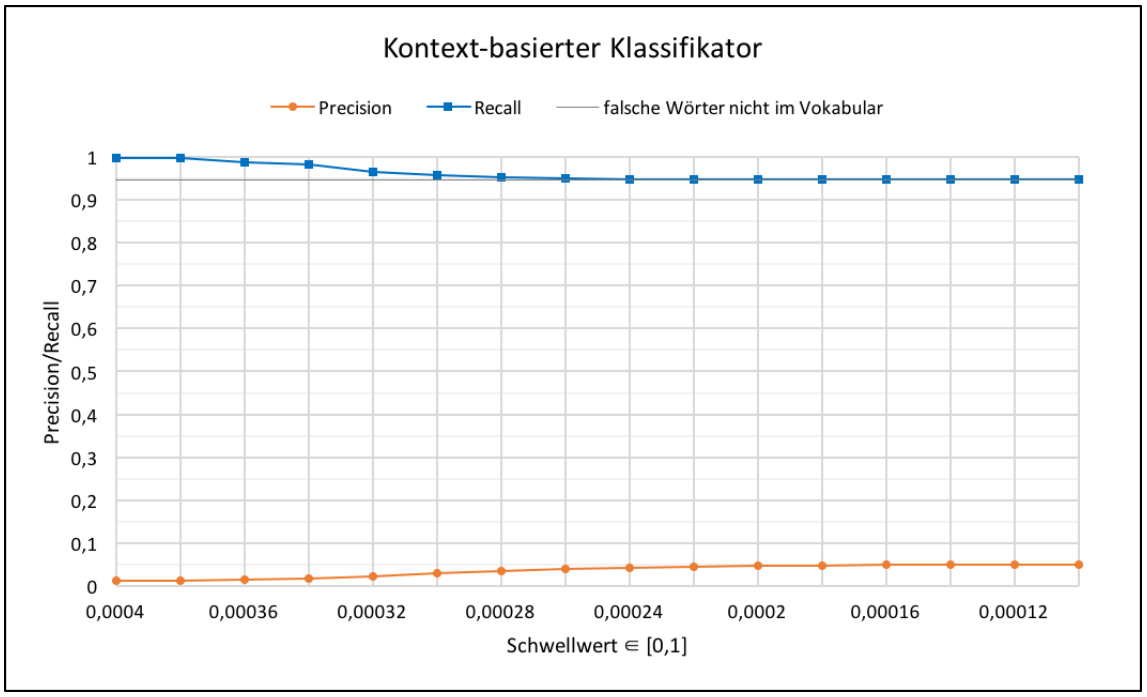

Abbildung 5.2: Diagramm mit Recall und Precision, Ergebnis 2, Kontext-basierter Klassifikator, Zeitungsartikel-Modell. Auch Wörter, die nicht im Vokabular enthalten sind, wurden bei der Berechnung berücksichtigt.

Tabelle 5.2: Tabelle mit weiteren Ausgabe-Kennzahlen, Ergebnis 2, Kontext-basierter Klassifikator, Zeitungsartikel-Modell. Auch Wörter, die nicht im Vokabular enthalten sind, wurden bei der Berechnung berücksichtigt.

Schwellwert $\in [\mathbf{0}, \mathbf{1}]$	True-Neg.	True-Pos.	davon nicht im Vokabular	False-Pos.	davon nicht im Vokabular	False-Neg.
0,00040	2534	827	785 (94,92%)	65477	15266 (23,32%)	2
0,00038	3915	827	785 (94,92%)	64096	15266 (23,82%)	2
0,00036	9734	820	785 (95,73%)	58277	15266 (26,20%)	9
0,00034	20204	815	785 (96,32%)	47807	15266 (31,93%)	14
0,00032	33287	801	785 (98%)	34724	15266 (43,96%)	28
0,00030	41246	795	785 (98,74%)	26765	15266 (57,04%)	34
0,00028	45684	790	785 (99,37%)	22327	15266 (68,37%)	39
0,00026	48455	787	785 (99,75%)	19556	15266 (78,06%)	42
0,00024	50232	786	785 (99,87%)	17779	15266 (85,87%)	43
0,00022	51326	785	785 (100%)	16685	15266 (91,50%)	44
0,00020	51998	785	785 (100%)	16013	15266 (95,34%)	44
0,00018	52378	785	785 (100%)	15633	15266 (97,65%)	44
0,00016	52594	785	785 (100%)	15417	15266 (99,02%)	44
0,00014	52691	785	785 (100%)	15320	15266 (99,65%)	44
0,00012	52727	785	785 (100%)	15284	15266 (99,88%)	44
0,00010	52738	785	785 (100%)	15273	15266 (99,95%)	44

Ergebnis 3: Kontext-basierter Klassifikator, Wikipedia-Modell

In Abbildung 5.3 und Tabelle 5.3, siehe rechts, sind die Daten zu Ergebnis 3 dargestellt. Bei diesem Ergebnis wurden vor der Datenerhebung alle Wörter, die nicht im Vokabular des Wikipedia-Modells enthalten sind, gelöscht. Es gibt also, anders wie bei Ergebnis 1, keine untere Schranke für den Recall.

Das Experiment wurde für mehrere Schwellwerte, von 10^{-3} bis 10^{-18}, wiederholt.

Der Recall sinkt mit kleiner werdendem Schwellwert gegen 0 und hat bei 10^{-13} Null erreicht. Anders als bei Ergebnis 1 steigt die Precision nicht sichtbar an und ist konstant fast 0.

Die Anzahl der True-Negatives steigt von 10^{-6} auf 10^{-7} stark an, um etwa 22 000 Wörter. Ab 10^{-13} ist die Steigung mit unter 100 Wörtern pro kleinerem Schwellwert sehr gering.

Prozentual an allen korrekten Wörtern gemessen, steigt die Anzahl der True-Negatives im selben Maß an, wie der Recall der fehlerhaften Wörter sinkt.

Ergebnis 4: Kontext-basierter Klassifikator, Zeitungsartikel-Modell

In Abbildung 5.4 und Tabelle 5.4, siehe Seite 36, sind die Daten zu Ergebnis 4 dargestellt. Bei diesem Ergebnis wurden vor der Datenerhebung alle Wörter, die nicht im Vokabular des Zeitungsartikel-Modells enthalten sind, gelöscht. Es gibt also, anders wie bei Ergebnis 2, keine untere Schranke für den Recall.

Das Experiment wurde für mehrere Schwellwerte, von 0,0004 bis 0,00027, wiederholt.

Das Gesamtbild ist dem von Ergebnis 3 sehr ähnlich. Der Recall sinkt mit kleiner werdendem Schwellwert gegen 0, die Precision ist sehr nahe 0. Ab einem Schwellwert von 0,0003 steigt die Precision sichtbar minimal an. Die maximale Precision wird mit 0,004 bei einem Schwellwert von 0,00028 erreicht. Das ist die höchste Precision von allen Ergebnissen, die mit dem Zeitungsartikel-Modell berechnet wurden.

Die Anzahl der True-Negatives steigt an, je kleiner der Schwellwert wird. Es gibt hier keinen großen Sprung in der Anzahl der True-Negatives, wie bei Ergebnis 3.

Prozentual an allen korrekten Wörtern gemessen, steigt die Anzahl der True-Negatives im selben Maß an, wie der Recall der fehlerhaften Wörter sinkt.

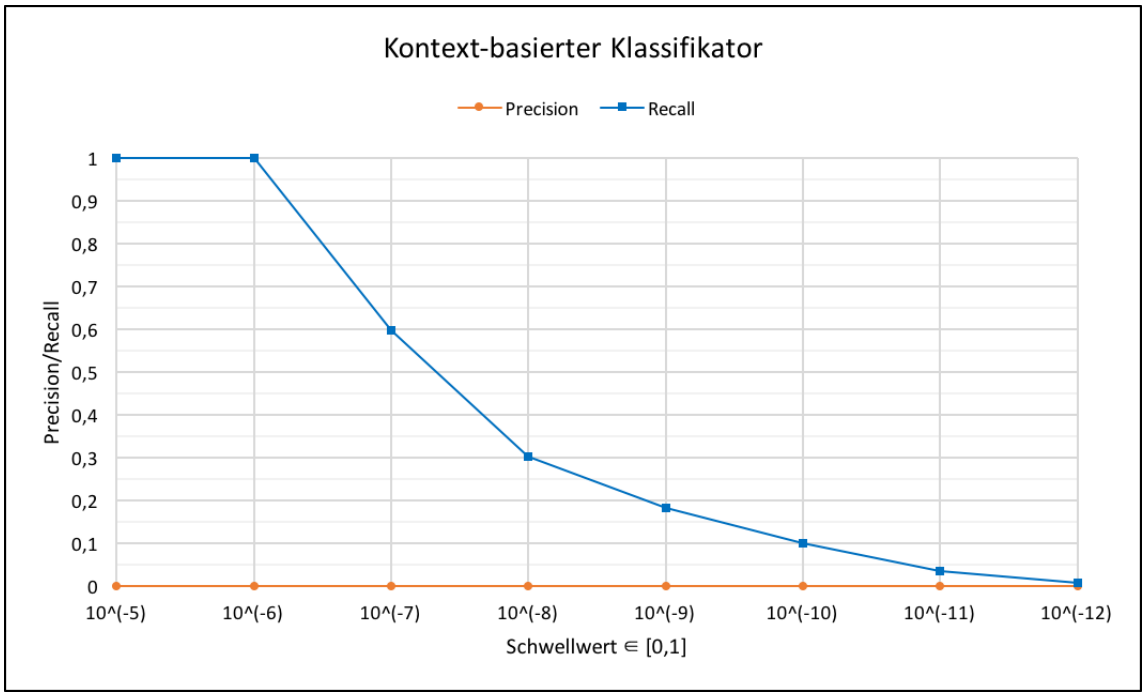

Abbildung 5.3: Diagramm mit Recall und Precision, Ergebnis 3, Kontext-basierter Klassifikator, Wikipedia-Modell.

Tabelle 5.3: Tabelle mit weiteren Ausgabe-Kennzahlen, Ergebnis 3, Kontext-basierter Klassifikator, Wikipedia-Modell.

Schwellwert $\in [\mathbf{0}, \mathbf{1}]$	True-Neg.	True-Pos.	False-Pos.	False-Neg.
$\mathbf{10^{-3}}$	2	109	55171	0
$\mathbf{10^{-4}}$	5	109	55168	0
$\mathbf{10^{-5}}$	55	109	55118	0
$\mathbf{10^{-6}}$	420	109	54753	0
$\mathbf{10^{-7}}$	22274	65	32899	44
$\mathbf{10^{-8}}$	34391	33	20782	76
$\mathbf{10^{-9}}$	42652	20	12521	89
$\mathbf{10^{-10}}$	49425	11	5748	98
$\mathbf{10^{-11}}$	52836	4	2337	105
$\mathbf{10^{-12}}$	54504	1	669	108
$\mathbf{10^{-13}}$	55022	0	151	109
$\mathbf{10^{-14}}$	55124	0	49	109
$\mathbf{10^{-15}}$	55157	0	16	109
$\mathbf{10^{-16}}$	55168	0	5	109
$\mathbf{10^{-17}}$	55170	0	3	109
$\mathbf{10^{-18}}$	55173	0	0	109

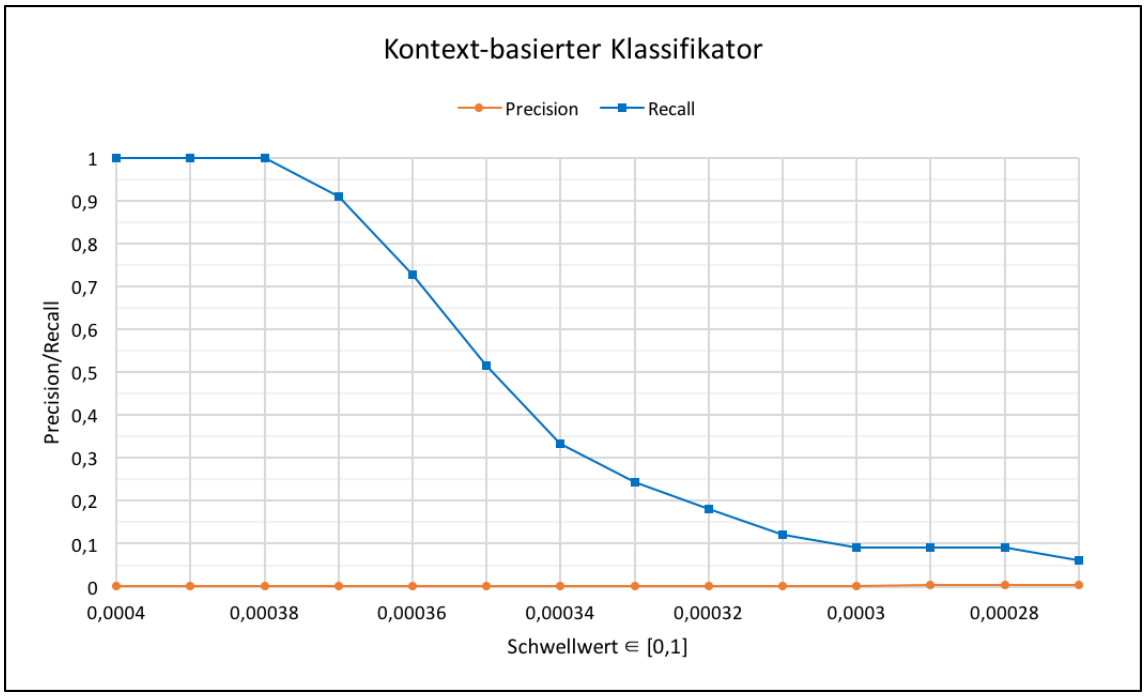

Abbildung 5.4: Diagramm mit Recall und Precision, Ergebnis 4, Kontext-basierter Klassifikator, Zeitungsartikel-Modell.

Tabelle 5.4: Tabelle mit weiteren Ausgabe-Kennzahlen, Ergebnis 4, Kontext-basierter Klassifikator, Zeitungsartikel-Modell.

Schwellwert $\in [\mathbf{0}, \mathbf{1}]$	True-Neg.	True-Pos.	False-Pos.	False-Neg.
0,00040	1988	33	40186	0
0,00039	2681	33	39493	0
0,00038	4080	33	38094	0
0,00037	7449	30	34725	3
0,00036	16017	24	26157	9
0,00035	25300	17	16874	16
0,00034	31879	11	10295	22
0,00033	35928	8	6246	25
0,00032	38431	6	3743	27
0,00031	39872	4	2302	29
0,00030	40679	3	1495	30
0,00029	41129	3	1045	30
0,00028	41429	3	745	30
0,00027	41650	2	524	31

Vorgänger-basierter Klassifikator

Ergebnis 5: Vorgänger-basierter Klassifikator, Wikipedia-Modell

In Abbildung 5.5 und Tabelle 5.5, siehe Seite 38, sind die Daten zu Ergebnis 5 dargestellt. Das Experiment wurde für die Schwellwerte zwischen 10^{-4} und 10^{-19} wiederholt.

Mit kleiner werdendem Schwellwert geht der Recall gegen 0. Die Precision steigt nicht sichtbar an und ist konstant fast 0. Bis auf die getesteten Schwellwerte unterscheiden sich diese Daten kaum von Ergebnis 3.

Die Anzahl der True-Negatives steigt mit kleiner werdendem Schwellwert kontinuierlich an. Einen großen Sprung von einem Schwellwert zu einem anderen wie bei Ergebnis 3 gibt es nicht.

Prozentual an allen korrekten Wörtern gemessen, steigt die Anzahl der True-Negatives im selben Maß an, wie der Recall der fehlerhaften Wörter sinkt.

Ergebnis 6: Vorgänger-basierter Klassifikator, Zeitungsartikel-Modell

In Abbildung 5.6 und Tabelle 5.6, siehe Seite 39, sind die Daten zu Ergebnis 6 dargestellt. Das Experiment wurde für die Schwellwerte zwischen 0,0004 und 0,00027 wiederholt.

Je kleiner der Schwellwert wird, desto näher geht der Recall gegen 0. Darin unterscheidet sich dieses Ergebnis nicht von den vorangegangenen Ergebnissen 3-5.

Die Precision ist ebenfalls konstant fast 0 und steigt nicht über 0,001. Damit unterscheidet sich dieses Ergebnis von Ergebnis 4, bei dem ebenfalls das Zeitungsartikel-Modell verwendet wurde. Die obere Schranke, die die Precision erreicht, ist hier nur $\frac{1}{4}$ der Schranke von Ergebnis 4.

Wie bei allen vorhergehenden Ergebnissen, steigt die Anzahl der True-Negatives mit kleiner werdendem Schwellwert an. Zwischen den Schwellwerten 0,00037 und 0,00036 gibt es einen größeren Unterschied von über 15 000 Wörtern, die der Klasse der True-Negatives zugeordnet werden.

Prozentual an allen korrekten Wörtern gemessen, steigt die Anzahl der True-Negatives gesamt gesehen im selben Maß an, wie der Recall der fehlerhaften Wörter sinkt. Im Detail ändern sich positive und negative Steigerung nicht immer um dieselbe Prozentzahl.

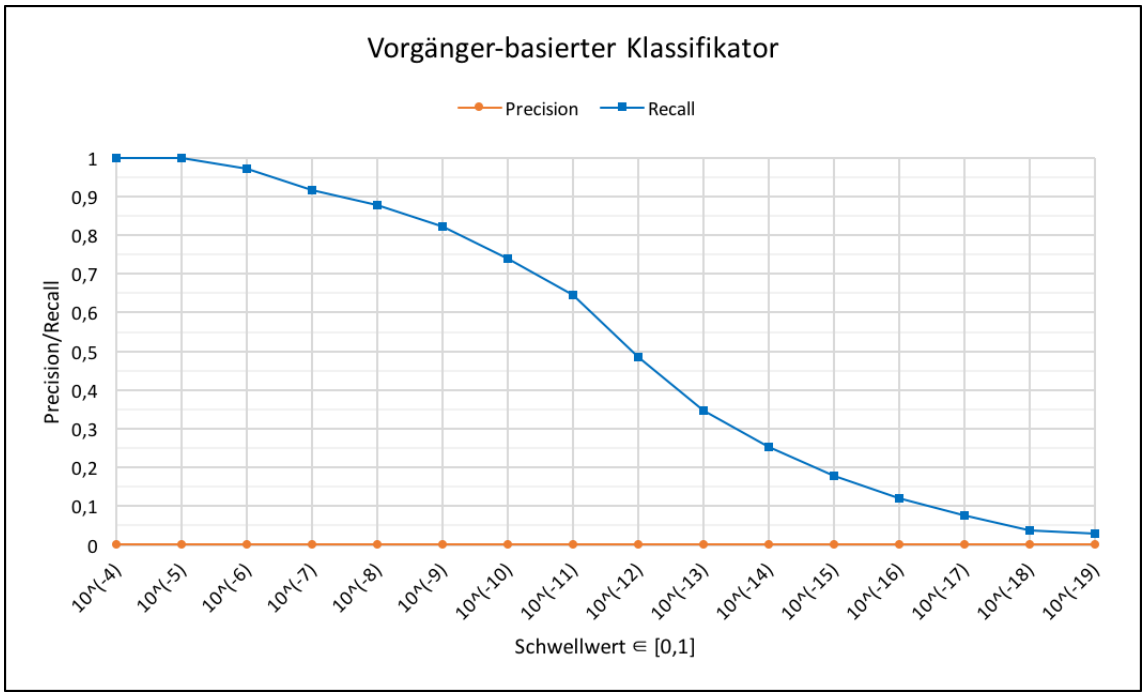

Abbildung 5.5: Diagramm mit Recall und Precision, Ergebnis 5, Vorgänger-basierter Klassifikator, Wikipedia-Modell.

Tabelle 5.5: Tabelle mit weiteren Ausgabe-Kennzahlen, Ergebnis 5, Vorgänger-basierter Klassifikator, Wikipedia-Modell.

Schwellwert $\in [\mathbf{0}, \mathbf{1}]$	True-Neg.	True-Pos.	False-Pos.	False-Neg.
$\mathbf{10^{-4}}$	6	107	54966	0
$\mathbf{10^{-5}}$	28	107	54944	0
$\mathbf{10^{-6}}$	2751	104	52221	3
$\mathbf{10^{-7}}$	5842	98	49130	9
$\mathbf{10^{-8}}$	8731	94	46241	13
$\mathbf{10^{-9}}$	13204	88	41768	19
$\mathbf{10^{-10}}$	17741	79	37231	28
$\mathbf{10^{-11}}$	23270	69	31702	38
$\mathbf{10^{-12}}$	29619	52	25353	55
$\mathbf{10^{-13}}$	36292	37	18680	70
$\mathbf{10^{-14}}$	42074	27	12898	80
$\mathbf{10^{-15}}$	45538	19	9434	88
$\mathbf{10^{-16}}$	47969	13	7003	94
$\mathbf{10^{-17}}$	50135	8	4837	99
$\mathbf{10^{-18}}$	51614	4	3358	103
$\mathbf{10^{-19}}$	52341	3	2631	104

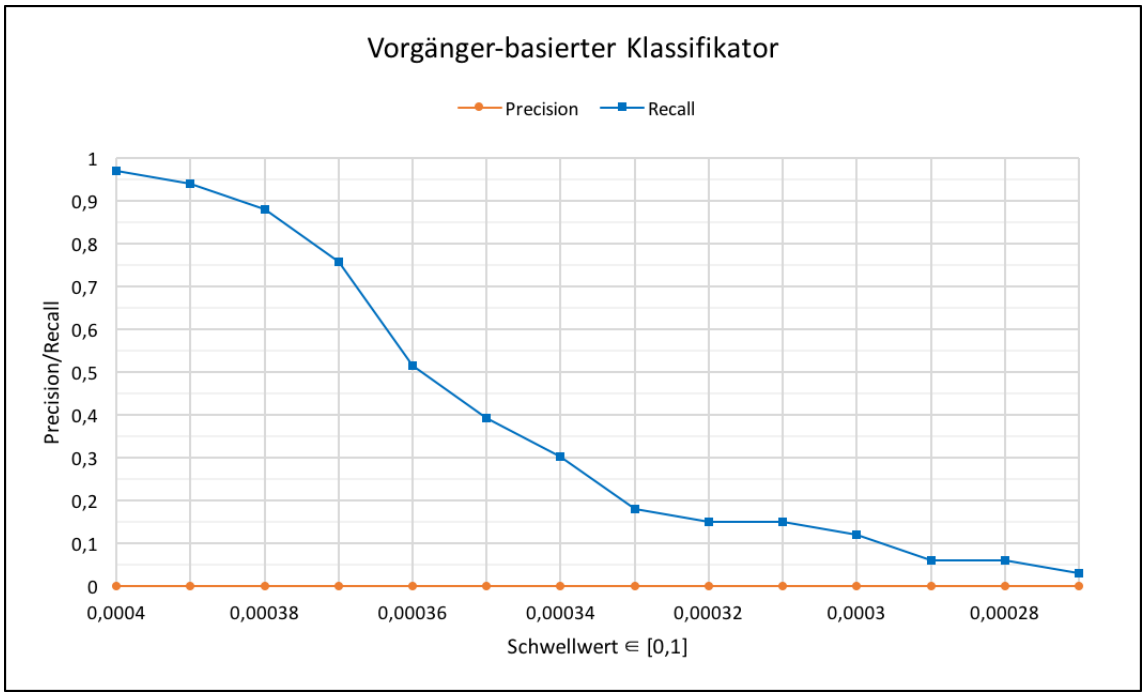

Abbildung 5.6: Diagramm mit Recall und Precision, Ergebnis 6, Vorgänger-basierter Klassifikator, Zeitungsartikel-Modell.

Tabelle 5.6: Tabelle mit weiteren Ausgabe-Kennzahlen, Ergebnis 6, Vorgänger-basierter Klassifikator, Zeitungsartikel-Modell.

Schwellwert $\in [\mathbf{0}, \mathbf{1}]$	True-Neg.	True-Pos.	False-Pos.	False-Neg.
0,00040	2452	32	39524	1
0,00039	3207	31	38769	2
0,00038	4699	29	37277	4
0,00037	8747	25	33229	8
0,00036	24433	17	17543	16
0,00035	29655	13	12321	20
0,00034	32603	10	9373	23
0,00033	34426	6	7550	27
0,00032	35915	5	6061	28
0,00031	36986	5	4990	28
0,00030	37826	4	4150	29
0,00029	38518	2	3458	31
0,00028	38979	2	2997	31
0,00027	39476	1	2500	32

Mittelvektor Klassifikator

Ergebnis 7: Mittelvektor Klassifikator, Wikipedia-Modell

In Abbildung 5.7 und Tabelle 5.7, siehe rechts, sind die Daten zu Ergebnis 7 dargestellt. Das Experiment wurde für die Schwellwerte zwischen 10^{-3} und 10^{-19} wiederholt.

Je kleiner der Schwellwert wird, desto stärker nähert sich der Recall 0 an, bis er bei 10^{-11} gleich 0 ist. Zu bemerken ist, dass der Recall bei diesem Ergebnis am schnellsten abfällt, betrachtet man die Ergebnisse 5 und 3, die ebenfalls mit dem Wikipedia-Modell berechnet wurden. Zwischen 10^{-5} und 10^{-7} fällt der Recall von fast 1 auf unter 0,1.

Eine entgegengesetzte Entwicklung zeigt sich bei der Betrachtung der True-Negatives. Je kleiner der Schwellwert wird, desto höher ist die Anzahl der True-Negatives. Zwischen 10^{-5} und 10^{-7} sind zwei große Sprünge in der Klassifizierung zu sehen: Zwischen 10^{-5} und 10^{-6} besteht ein Unterschied von über 27 000 True-Negatives und zwischen 10^{-6} und 10^{-7} ein Unterschied von über 25 000 True-Negatives.

Die Precision ist sehr nahe 0, steigt jedoch ab 10^{-7} sichtbar an. Bei 10^{-9} steigt sie auf 0,011 und erreicht damit die höchste Precision von allen Ergebnissen.

Prozentual an allen korrekten Wörtern gemessen, steigt die Anzahl der True-Negatives im selben Maß an, wie der Recall der fehlerhaften Wörter sinkt.

Ergebnis 8: Mittelvektor Klassifikator, Zeitungsartikel-Modell

In Abbildung 5.8 und Tabelle 5.8, siehe Seite 42, sind die Daten zu Ergebnis 8 dargestellt. Das Experiment wurde für die Schwellwerte zwischen 0,0004 und 0,00027 wiederholt.

Mit kleiner werdendem Schwellwert geht der Recall, wie bei den Ergebnissen 3-7, gegen 0. Die Precision ist ebenfalls konstant fast 0 und steigt nicht merklich an. Sie steigt auf maximal 0,0036 bei einem Schwellwert von 0,00028 und weicht von der maximalen Precision bei Ergebnis 4 um nur 0,0004 ab.

Analog zu den bisherigen Ergebnissen steigt die Anzahl der True-Positives mit kleiner werdendem Schwellwert.

Prozentual an allen korrekten Wörtern gemessen, steigt die Anzahl der True-Negatives gesamt gesehen im selben Maß an, wie der Recall der fehlerhaften Wörter sinkt. Im Detail stimmen die positive und negative Steigung nicht immer überein. Vor allem bei der Wahl des Schwellwerts zwischen 0,0004 und 0,00037 ist eine Abweichung deutlich sichtbar; die Anzahl der True-Positives bleibt gleich, die Anzahl der True-Negatives nimmt deutlich zu. Trotzdem ist der Anteil der fehlerhaften Wörter in der Klasse der Schreib- und Digitalisierungsfehler mit unter 0,1% sehr gering.

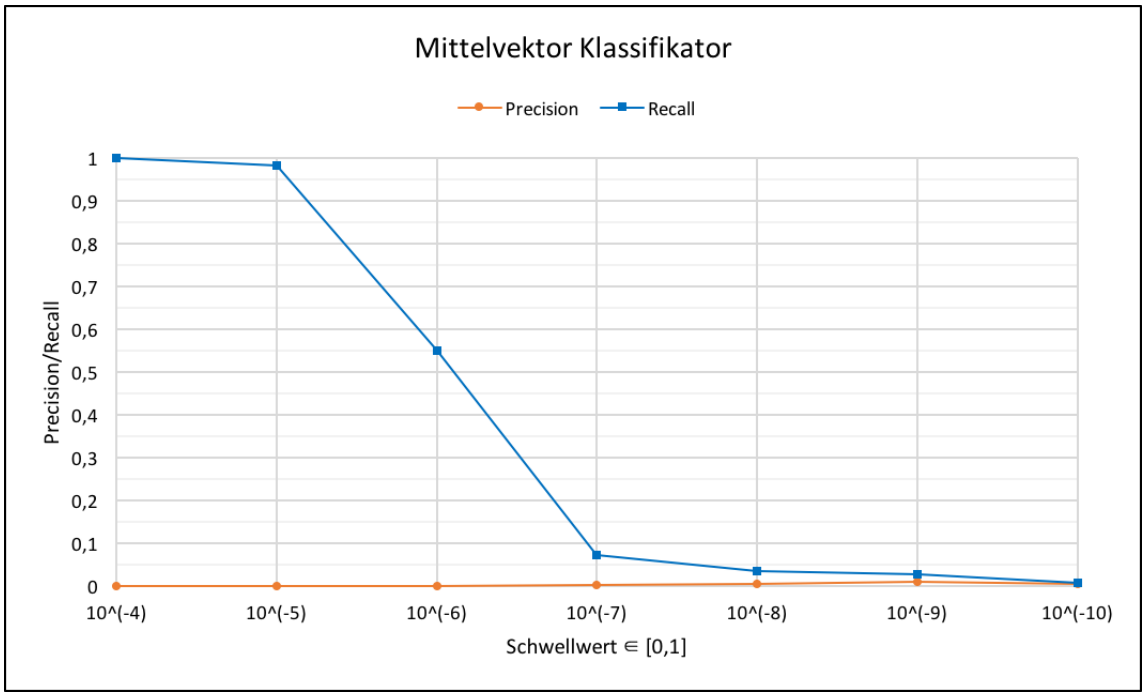

Abbildung 5.7: Diagramm mit Recall und Precision, Ergebnis 7, Mittelvektor Klassifikator, Wikipedia-Modell.

Tabelle 5.7: Tabelle mit weiteren Ausgabe-Kennzahlen, Ergebnis 7, Mittelvektor Klassifikator, Wikipedia-Modell.

Schwellwert $\in [\mathbf{0}, \mathbf{1}]$	True-Neg.	True-Pos.	False-Pos.	False-Neg.
$\mathbf{10^{-3}}$	0	109	55173	0
$\mathbf{10^{-4}}$	13	109	55160	0
$\mathbf{10^{-5}}$	396	107	54777	2
$\mathbf{10^{-6}}$	27789	60	27384	49
$\mathbf{10^{-7}}$	53193	8	1980	101
$\mathbf{10^{-8}}$	54596	4	577	105
$\mathbf{10^{-9}}$	54904	3	269	106
$\mathbf{10^{-10}}$	55021	1	152	108
$\mathbf{10^{-11}}$	55094	0	79	109
$\mathbf{10^{-12}}$	55127	0	46	109
$\mathbf{10^{-13}}$	55155	0	18	109
$\mathbf{10^{-14}}$	55162	0	11	109
$\mathbf{10^{-15}}$	55168	0	5	109
$\mathbf{10^{-16}}$	55171	0	2	109
$\mathbf{10^{-17}}$	55172	0	1	109
$\mathbf{10^{-18}}$	55172	0	1	109
$\mathbf{10^{-19}}$	55173	0	0	109

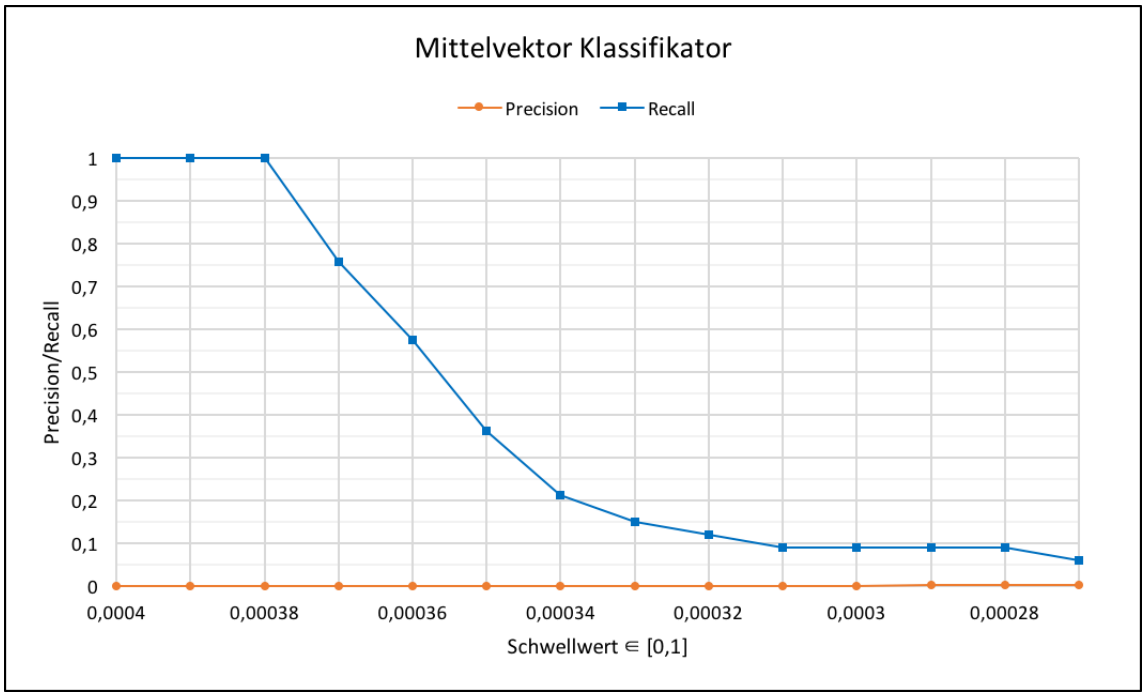

Abbildung 5.8: Diagramm mit Recall und Precision, Ergebnis 8, Mittelvektor Klassifikator, Zeitungsartikel-Modell.

Tabelle 5.8: Tabelle mit weiteren Ausgabe-Kennzahlen, Ergebnis 8, Mittelvektor Klassifikator, Zeitungsartikel-Modell.

Schwellwert $\in [\mathbf{0}, \mathbf{1}]$	True-Neg.	True-Pos.	False-Pos.	False-Neg.
0,00040	2424	33	39750	0
0,00039	3592	33	38582	0
0,00038	6241	33	35933	0
0,00037	11915	25	30259	8
0,00036	23140	19	19034	14
0,00035	30632	12	11542	21
0,00034	34913	7	7261	26
0,00033	37653	5	4521	28
0,00032	39299	4	2875	29
0,00031	40261	3	1913	30
0,00030	40777	3	1397	30
0,00029	41107	3	1067	30
0,00028	41345	3	829	30
0,00027	41530	2	644	31

5.1.2 Wortvektorbasierte Klassifikatoren

Kosinus-Ähnlichkeit Klassifikator

Ergebnis 9: Kosinus-Ähnlichkeit Klassifikator, Wikipedia-Modell

In Abbildung 5.9 und Tabelle 5.9, siehe Seite 44, sind die Daten zu Ergebnis 9 dargestellt. Das Experiment wurde für die Schwellwerte zwischen 0,1 und 0,9 wiederholt.

Mit kleiner werdendem Schwellwert geht der Recall gegen 0. Bei einem Schwellwert von 0,3 ist der Recall gleich 0. Es gibt einen größeren Sprung zwischen 0,6 und 0,4: Der Recall sinkt von knapp 0,9 auf unter 0,1.

Ein umgekehrtes Bild zeigt sich bei der Betrachtung der True-Negatives. Je kleiner der Schwellwert wird, desto mehr Wörter sind in der Klasse der True-Neagtives. Hier gibt es ebenfalls zwei große Sprünge zwischen 0,6 und 0,4: Zwischen 0,6 und 0,5 besteht ein Unterschied von 20 000 True-Negatives und zwischen 0,5 und 0,4 ein Unterschied von knapp 18 000 Wörtern.

Gleich wie bei den Ergebnissen 3-8 ist die Precision konstant nahe 0. Sie steigt nicht sichtbar an.

Prozentual an allen korrekten Wörtern gemessen, steigt die Anzahl der True-Negatives bis auf kleine Abweichungen im selben Maß an, wie der Recall der fehlerhaften Wörter sinkt.

Ergebnis 10: Kosinus-Ähnlichkeit Klassifikator, Zeitungsartikel-Modell

In Abbildung 5.10 und Tabelle 5.10, siehe Seite 45, sind die Daten zu Ergebnis 10 dargestellt. Analog zu Ergebnis 9 wurde das Experiment für die Schwellwerte zwischen 0,1 und 0,9 wiederholt.

Der Recall geht mit sinkendem Schwellwert gegen 0. Ab einem Schwellwert von 0,3 ist der Recall gleich 0, gleich wie bei Ergebnis 9. Es gibt jedoch keine auffälligen Sprünge zwischen verschiedenen Schwellwerten, wie das bei Ergebnis 9 der Fall ist. Die Precision ist ebenfalls konstant fast 0.

Je kleiner der Schwellwert wird, desto höher ist die Anzahl der True-Negatives. Das entspricht demselben Bild wie bei allen bisherigen Ergebnissen.

Prozentual an allen korrekten Wörtern gemessen, steigt die Anzahl der True-Negatives bis auf kleine Abweichungen im selben Maß an, wie der Recall der fehlerhaften Wörter sinkt.

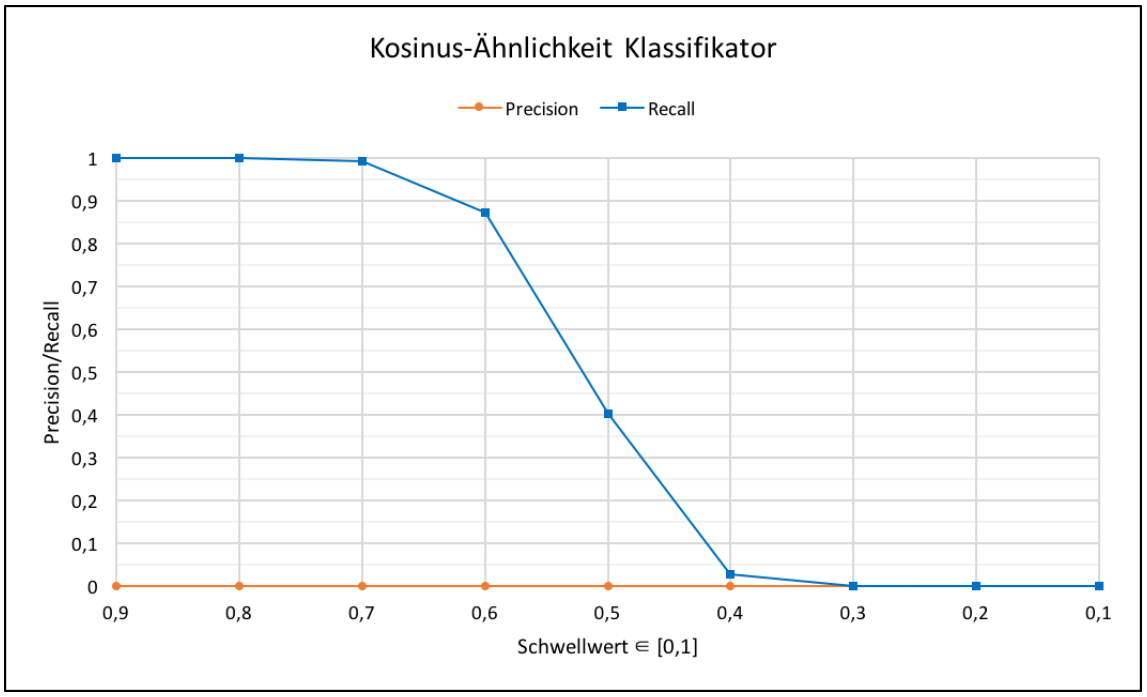

Abbildung 5.9: Diagramm mit Recall und Precision, Ergebnis 9, Kosinus-Ähnlichkeit Klassifikator, Wikipedia-Modell.

Tabelle 5.9: Tabelle mit weiteren Ausgabe-Kennzahlen, Ergebnis 9, Kosinus-Ähnlichkeit Klassifikator, Wikipedia-Modell.

Schwellwert $\in [\mathbf{0}, \mathbf{1}]$	True-Neg.	True-Pos.	False-Pos.	False-Neg.
0,9	72	109	55101	0
0,8	397	109	54776	0
0,7	1927	108	53246	1
0,6	9795	95	45378	14
0,5	29819	44	25354	65
0,4	47622	3	7551	106
0,3	54309	0	864	109
0,2	55164	0	9	109
0,1	55173	0	0	109

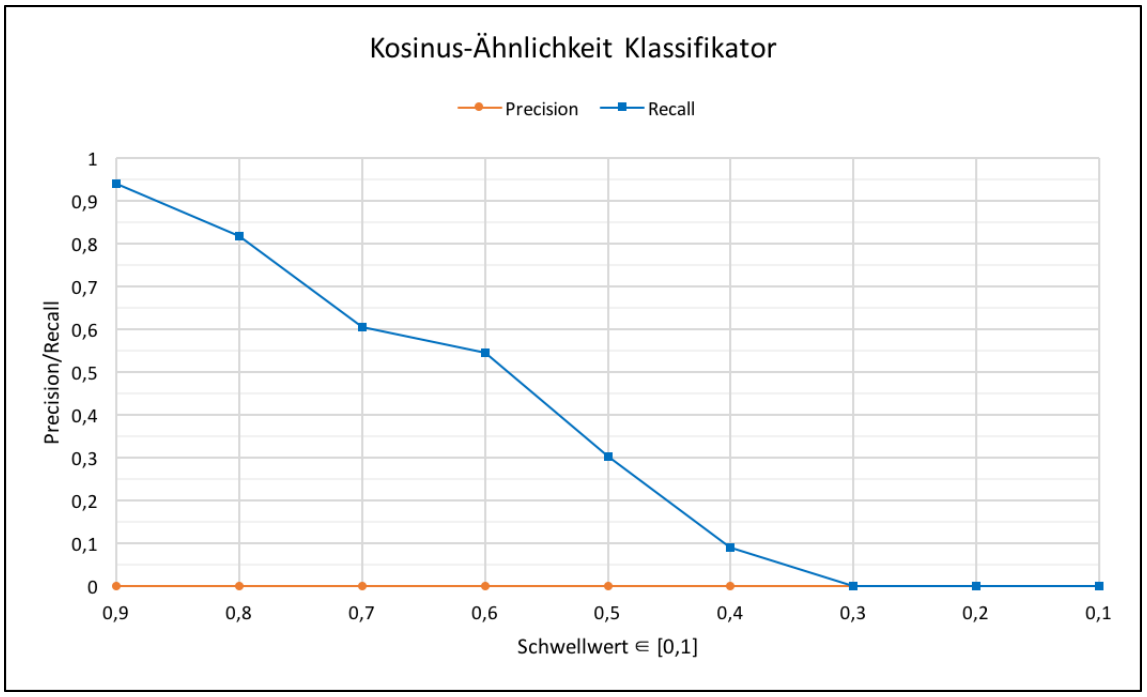

Abbildung 5.10: Diagramm mit Recall und Precision, Ergebnis 10, Kosinus-Ähnlichkeit Klassifikator, Zeitungsartikel-Modell.

Tabelle 5.10: Tabelle mit weiteren Ausgabe-Kennzahlen, Ergebnis 10, Kosinus-Ähnlichkeit Klassifikator, Zeitungsartikel-Modell.

Schwellwert $\in [\mathbf{0}, \mathbf{1}]$	True-Neg.	True-Pos.	False-Pos.	False-Neg.
0,9	2789	31	39385	2
0,8	10592	27	31582	6
0,7	19495	20	22679	13
0,6	28131	18	14043	15
0,5	35099	10	7075	23
0,4	39299	3	2875	30
0,3	41176	0	998	33
0,2	41919	0	255	33
0,1	42155	0	19	33

Euklidische Distanz Klassifikator

Ergebnis 11: Euklidische Distanz Klassifikator, Wikipedia-Modell

In Abbildung 5.11 und Tabelle 5.11, siehe rechts, sind die Daten zu Ergebnis 11 dargestellt. Das Experiment wurde für die Schwellwerte zwischen 1 und 0,65 wiederholt.

Der Recall geht mit sinkendem Schwellwert gegen 0. Ab einem Schwellwert von 0,7 ist der Recall gleich 0. Ein größerer Sprung findet zwischen 0,95 und 0,9 statt: Der Recall sinkt von 0,8 auf 0,4.

Das entgegengesetzte Bild zeigt sich bei der Betrachtung der True-Negatives. Je kleiner der Schwellwert wird, desto mehr Wörter sind in der Klasse der True-Negatives. Ein größerer Sprung findet zwischen 0,9 und 0,85 statt – der Unterschied in der Anzahl der True-Negatives beträgt knapp 23 000.

Die Precision ist, gleich wie bei den bisherigen Ergebnissen 3-10, konstant nahe 0. Sie steigt nicht sichtbar an.

Prozentual an allen korrekten Wörtern gemessen, steigt die Anzahl der True-Negatives etwas weniger stark an, wie der Recall der fehlerhaften Wörter sinkt.

Ergebnis 12: Euklidische Distanz Klassifikator, Zeitungsartikel-Modell

In Abbildung 5.12 und Tabelle 5.12, siehe Seite 48, sind die Daten zu Ergebnis 12 dargestellt. Das Experiment wurde für die Schwellwerte zwischen 1 und 0,75 wiederholt.

Je kleiner der Schwellwert wird, desto näher geht der Recall gegen 0. Bei einem Schwellwert von 0,85 ist der Recall gleich 0. Es ist ein größerer Sprung im Abfall des Recalls zu beobachten: Zwischen den Schwellwerten 0,975 und 0,925 sinkt der Recall von über 0,95 auf 0,15 ab.

Mit kleiner werdendem Schwellwert steigt die Anzahl der True-Negatives kontinuierlich an.

Die Precision ist, ähnlich wie bei den Ergebnissen 3-11, konstant nahe 0.

Prozentual an allen korrekten Wörtern gemessen, steigt die Anzahl der True-Negatives etwas weniger stark an, wie der Recall der fehlerhaften Wörter sinkt.

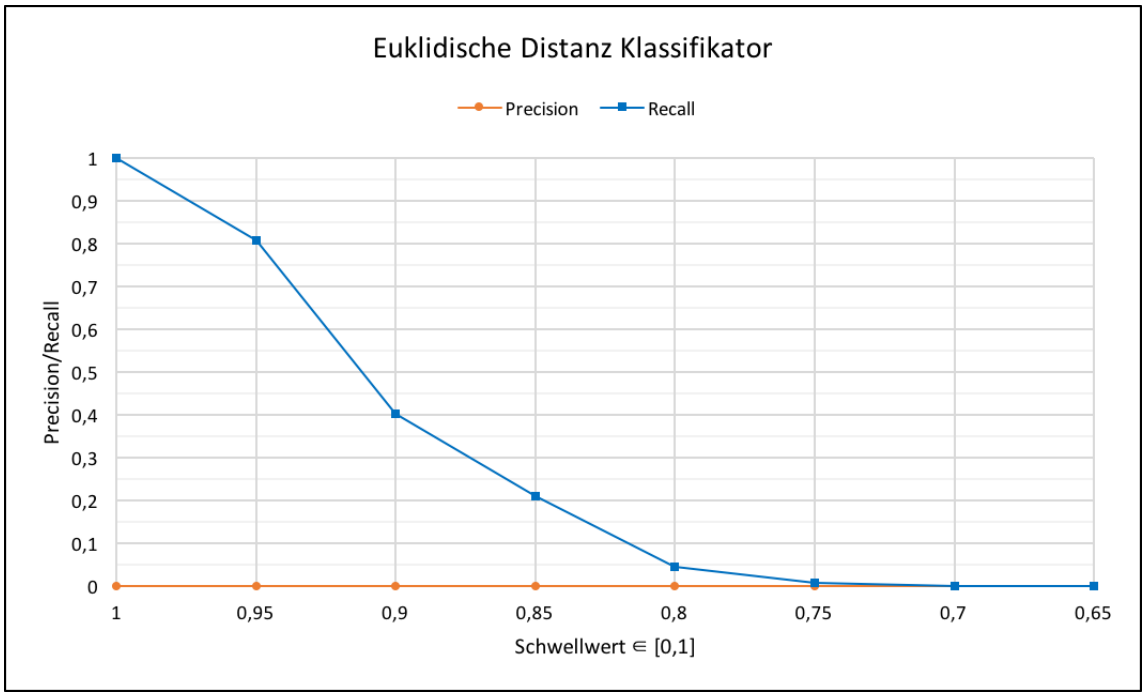

Abbildung 5.11: Diagramm mit Recall und Precision, Ergebnis 11, Kosinus-Abstand Klassifikator, Wikipedia-Modell.

Tabelle 5.11: Tabelle mit weiteren Ausgabe-Kennzahlen, Ergebnis 11, Euklidische Distanz Klassifikator, Wikipedia-Modell.

Schwellwert $\in [\mathbf{0}, \mathbf{1}]$	True-Neg.	True-Pos.	False-Pos.	False-Neg.
1,00	0	109	55173	0
0,95	1423	88	53750	21
0,90	12417	44	42756	65
0,85	35375	23	19798	86
0,80	51040	5	4133	104
0,75	54718	1	455	108
0,70	55144	0	29	109
0,65	55173	0	0	109

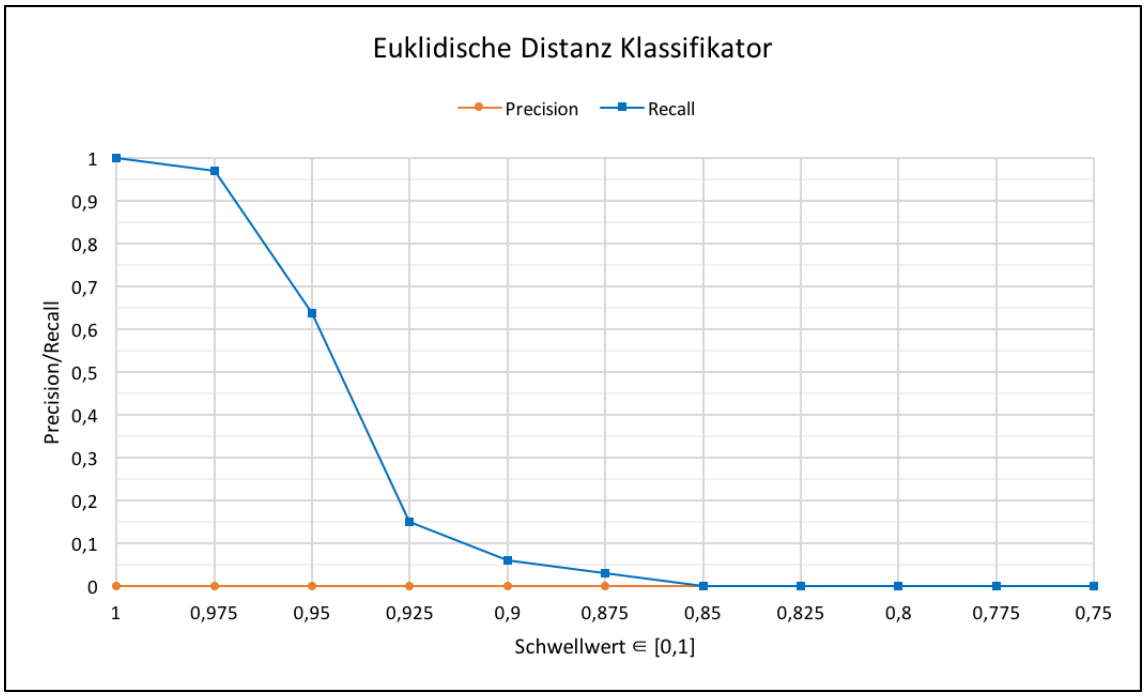

Abbildung 5.12: Diagramm mit Recall und Precision, Ergebnis 12, Kosinus-Abstand Klassifikator, Zeitungsartikel-Modell.

Tabelle 5.12: Tabelle mit weiteren Ausgabe-Kennzahlen, Ergebnis 12, Euklidische Distanz Klassifikator, Zeitungsartikel-Modell.

Schwellwert $\in [\mathbf{0}, \mathbf{1}]$	True-Neg.	True-Pos.	False-Pos.	False-Neg.
1	0	33	42174	0
0,975	1042	32	41132	1
0,950	12939	21	29235	12
0,925	24407	5	17767	28
0,900	31340	2	10834	31
0,875	35162	1	7012	32
0,850	38741	0	3433	33
0,825	40584	0	1590	33
0,800	41772	0	402	33
0,775	42171	0	3	33
0,750	42174	0	0	33

5.2 Diskussion

In diesem Abschnitt werden die im vorherigen Abschnitt 5.1 dargestellten Ergebnisse diskutiert und auf ihre Anwendbarkeit zur Fehlererkennung in Texten analysiert. Zunächst wird auf diejenigen Wörter eingegangen, die nicht im Vokabular eines CBOW-Modells enthalten sind. Danach folgt die Diskussion der Ergebnisse für die in Kapitel 3 vorgestellten Klassifikatoren.

5.2.1 Wörter, die nicht im Vokabular enthalten sind

Dieser Abschnitt untersucht den Teil der Ergebnisse, der nicht mit den Wortvektoren sondern mit dem Vokabular des CBOW-Modells arbeitet. Konkret werden diejenigen Wörter diskutiert, die nicht im Vokabular eines Modells enthalten sind. Die Daten ergeben sich also aus der Abwesenheit eines Wortvektors für das konkrete Wort. Diese Wörter sind für jeden Klassifikator gleich.

Die Ergebnisse 1 und 2 enthalten, im Gegensatz zu den anderen in Abschnitt 5.1 dargestellten Ergebnissen, auch diejenigen Wörter, die nicht im Vokabular des jeweiligen Modells enthalten sind.

Bei Verwendung des Wikipedia-Modells sind 85,5% der fehlerhaften Wörter aus den Validierungstexten nicht im Vokabular enthalten. Das heißt, durch eine Wörterbuchsuche im Vokabular des Wikipedia-Modells können um die 85% der fehlerhaften Wörter identifiziert werden.
Betrachtet man die Zahlen in Abschnitt 4.1 – *Validierungsdaten*, sieht man, dass nur etwa 12% aller fehlerhaften Wörter existierende Wörter darstellen. Daraus könnte man schließen, dass maximal 88% der fehlerhaften Wörter nicht im Vokabular des Modells enthalten sein können. Dieser Schluss ist jedoch aus folgenden Gründen nicht korrekt: Das Vokabular des Wikipedia-Modells enthält zusätzlich zu Wörtern auch einzelne Buchstaben und Punktuationszeichen. Diese werden in Abschnitt 4.1 nicht zu der Klasse der existierenden Wörter gezählt. Außerdem enthält das Vokabular Abkürzungen wie `Ms`, das aus dem Englischen stammen dürfte, `cs` und Eigennamen wie `Billa`. Alle im vorigen Satz genannten Beispiele sind fehlerhafte Digitalisierungen im Validierungsdatensatz.
3,3% der korrekten Wörter sind nicht im Vokabular des Wikipedia-Modells enthalten. Das bedeutet, dass diese Wörter immer als fehlerhaft klassifiziert werden, obwohl sie korrekt wären.
Bei Betrachtung der absoluten Wortanzahlen sind etwa 3 mal so viele korrekte Wörter nicht im Vokabular des Wikipedia-Modells enthalten wie fehlerhafte.

Bei Verwendung des Zeitungsartikel-Modells sind 94,9% der fehlerhaften Wörter aus den Validierungstexten nicht im Vokabular enthalten. Das heißt, durch eine Wörterbuchsuche im Vokabular des Zeitungsartikel-Modells können um die 95% der fehlerhaften Wörter identifiziert werden. Von den korrekten Wörtern sind 15 266 Wörter nicht im Vokabular des Modells enthalten, was 22,4% aller korrekten Wörter entspricht. Das heißt, diese Wörter werden immer als fehlerhaft klassifiziert, obwohl sie korrekt wären.
Bei Betrachtung der absoluten Wortanzahlen sind etwa 20 mal so viele korrekte Wörter nicht im Vokabular des Zeitungsartikel-Modells enthalten wie fehlerhafte.

Dieses Ungleichgewicht spricht deutlich gegen die Nutzung einer Wörterbuchsuche mit dem Vokabular des Zeitungsartikel-Modells.

Bei immer kleiner werdenden Schwellwerten gibt es einen Schwellwert, ab dem jeder Klassifikator alle Wörter als korrekt klassifiziert, unabhängig davon, ob das Wort tatsächlich korrekt ist oder nicht. Ausgenommen sind diejenigen Wörter, die nicht im Vokabular des jeweiligen Modells enthalten sind. Die Precision steigt dann von fast 0, da die Anzahl der False-Positives sehr hoch ist, auf bis zu 0,2383 beim Wikipedia-Modell und 0,0485 beim Zeitungsartikel-Modell. Es findet also einerseits eine Umklassifizierung der False-Positives in True-Negatives statt. Das heißt, korrekte Wörter werden als solche erkannt. Andererseits findet eine Umklassifizierung der True-Positives zu False-Negatives statt, was bedeutet, dass fehlerhafte Wörter nicht mehr als solche erkannt werden.

Aus obiger Darstellung kann man schließen, dass sich das Word2Vec-Modell zur Identifizierung von fehlerhaften Wörtern in den gegebenen Validierungstexten eignen kann.
Soll das Modell zur Fehlerkorrektur verwendet werden, dann kann das Wikipedia-Modell eine akzeptable, aber bei Weitem nicht optimale, Lösung darstellen. Das Zeitungsartikel-Modell hingegen ist nicht für eine Anwendung zu empfehlen. Die Anzahl der korrekten Wörter, die dann bei einer Korrektur manuell überprüft werden müssten, ist um ein Vielfaches höher als die Anzahl der tatsächlich fehlerhaften Wörter.

Im nächsten Abschnitt werden die Ergebnisse zu den in Kapitel 3 vorgestellten Klassifikatoren analysiert und interpretiert. Die Ergebnisse mit der *Parameter-Standard-Einstellung*, siehe Anfang Abschnitt 5.1, bilden die Grundlage der folgenden Diskussion. Dabei soll u.a. geklärt werden, ob sich die CBOW-Wortvektoren zur Fehlererkennung in Texten eignen.

5.2.2 Diskussion der Klassifikatoren

Die folgende Diskussion basiert auf den Ergebnissen 3-12, siehe Abschnitt 5.1.

Wie man anhand der Daten in Abschnitt 5.1 sieht, findet eine Umklassifizierung der Wörter statt, wenn sich der Schwellwert ändert.

Je kleiner der Schwellwert wird, desto weniger Wörter mit Schreib- bzw. Digitalisierungsfehler werden als solche klassifiziert. Gleichzeitig steigt die Anzahl der True-Negatives an, je kleiner der Schwellwert wird.

Prozentual an allen korrekten Wörtern gemessen, steigt die Anzahl der True-Negatives bei Verringerung des Schwellwerts im selben Maß an, wie der Recall der korrekten Wörter sinkt. Das gilt für alle Klassifikatoren. In manchen Fällen stimmen die positive und negative Steigung nicht ganz überein.
Dies ist beim Vorgänger-basierten Klassifikator zusammen mit dem Zeitungsartikel-Modell der Fall, siehe Ergebnis 6. Die Abweichungen sind gering und gleichen sich mit Verkleinerung des Schwellwerts wieder aus.
Bei der Verwendung des Mittelvektor Klassifikators mit dem Zeitungsartikel-Modell

ist die Abweichung auffälliger, siehe Ergebnis 8. Bei der Wahl des Schwellwerts zwischen 0,0004 und 0,00037 bleibt die Anzahl der True-Positives gleich, die Anzahl der True-Negatives nimmt deutlich zu. Trotzdem ist der Anteil der fehlerhaften Wörter in der Klasse der Schreib- und Digitalisierungsfehler mit unter 0,1% sehr gering. Der Unterschied in der Steigung kann also vernachlässigt werden.
Bei Anwendung des Euklidische Distanz Klassifikators, siehe Ergebnisse 11 und 12, ergeben sich ebenfalls Unterschiede in der Steigung. Konkret sinkt der Recall bei höheren Schwellwerten etwas schneller als die Anzahl der True-Negatives steigt. Als Menge der Schwellwerte wird hier jenes Intervall verwendet, in dem eine Umklassifizierung der Wörter statt findet.
Diese Daten implizieren, dass die Wörter in etwa gleichverteilt umklassifiziert werden. Daraus lässt sich schließen, dass die mit diesen Methoden ermittelten Scores keinen Unterschied zwischen korrekten und nicht korrekten Wörtern darstellen.

Diese Aussage wird durch die folgende Analyse ebenfalls bestätigt. Wenn alle Wörter mit Schreib- bzw. Digitalisierungsfehler als solche erkannt werden, werden alle korrekten Wörter als fehlerhaft klassifiziert. Wenn umgekehrt alle korrekten Wörter als korrekt klassifiziert werden, werden auch alle fehlerhaften Wörter als korrekt klassifiziert.
Bei Anwendung des Mittelvektor-Klassifikators und des Zeitungsartikel-Modells, siehe Ergebnis 8, gilt diese Analyse in abgeschwächter Form. Wenn so gut wie alle Wörter mit Schreib- bzw. Digitalisierungsfehler als solche erkannt werden, werden 6 241 korrekte Wörter als korrekt klassifiziert. Das entspricht 14,8% der korrekten Wörter.
Aus diesen Daten könnte man schließen, dass der Mittelvektor Klassifikator korrekte und nicht korrekte Wörter unterscheiden kann. Jedoch sprechen mehr Argumente gegen diesen Schluss als dafür: zum Einen das am Anfang dieser Diskussion beschriebene Verhältnis der prozentualen Steigerung der korrekten und fehlerhaften Wörter, zum Anderen die im nächsten Absatz folgende Analyse zur Precision.

Bei der Betrachtung der Precision ergibt sich ein ähnliches Bild wie in den vorhergegangenen Absätzen. Die Precision ist bei allen Ergebnissen konstant fast 0; die maximal erreichte Precision ist beim Wikipedia-Modell 0,011 bei Ergebnis 7, beim Zeitungsartikel-Modell 0,004 bei Ergebnis 4. Das heißt, maximal 1,1% der Wörter aus der Klasse der Schreib- bzw. Digitalisierungsfehler sind tatsächlich solche. Auch das spricht dafür, dass die berechneten Scores kein Kriterium zur Klassifizierung der Wörter in fehlerhafte und korrekte Wörter darstellen.
Ein Grund für die niedrige Precision ist die sehr hohe Anzahl der korrekten Wörter im Verhältnis zu den fehlerhaften Wörtern. Bei Verwendung des Wikipedia-Modells sind insgesamt 506 mal mehr Wörter aus den Validierungstexten korrekt als fehlerhaft. Dort sind insgesamt nur 109 Wörter fehlerhaft. Bei Verwendung des Zeitungsartikel-Modells sind es 1 278 mal mehr korrekte Wörter als fehlerhafte. Dort sind nur 33 Wörter fehlerhaft.
Der Unterschied in der Anzahl der fehlerhaften Wörter ergibt sich aus den unterschiedlichen Vokabulargrößen. Beim Zeitungsartikel-Modell sind weit mehr fehlerhafte Wörter nicht im Vokabular enthalten als beim Wikipedia-Modell.

Insgesamt folgt aus obiger Darlegung, dass diese Klassifikatoren nicht zur Erkennung fehlerhafter Wörter in einem Text geeignet sind. Sie unterscheiden nicht zwischen

korrekten und fehlerhaften Wörtern.
Bei den meisten Klassifikatoren konnte kein nennenswerter Unterschied zwischen den Ergebnissen bei Verwendung des Wikipedia-Modells und des Zeitungsartikel-Modells erzielt werden.

Beim Mittelvektor Klassifikator haben sich Unterschiede bei der Verwendung eines domänenspezifischen und eines domänenunabhängigen Modells ergeben. Die Verwendung des Zeitungsartikel-Modells hat etwas bessere Ergebnisse geliefert als die Verwendung des Wikipedia-Modells, wenn alle fehlerhaften Wörter erkannt werden. Das Wikipedia-Modell hat eine fast doppelt so hohe maximale Precision wie das Zeitungsartikel-Modell, was für eine akkuratere Klassifikation durch das Wikipedia-Modell spricht. Beide Precisions sind unter 0,01. Trotz dieser Unterschiede ist das Gesamtbild bei beiden Modellen sehr ähnlich. Es kann folglich keine Aussage darüber getroffen werden, welches Modell insgesamt bessere Ergebnisse liefert.

Beim Euklidische Distanz Klassifikator ist der Abstand zwischen dem mittleren Vektor und dem Wortvektor interessanterweise immer kleiner $\frac{1}{2} \cdot N$, wobei N die Dimension des Vektorraums bezeichnet. Beim Zeitungsartikel-Modell sind alle Distanzen kleiner $0,23 \cdot N$, beim Wikipedia-Modell kleiner $0,34 \cdot N$.

Der euklidische Abstand zwischen dem Wortvektor des einzelnen Kontextworts und dem Wort, für das der Score berechnet werden soll, ist bei allen Wörtern in den Evaluierungstexten ebenfalls kleiner $\frac{1}{2} \cdot N$. Beim Zeitungsartikel-Modell sind alle Distanzen kleiner $0,3 \cdot N$, beim Wikipedia-Modell $0,48 \cdot N$.

Daraus kann man schließen, dass die Wortvektoren im Vektorraum in einem begrenzten Sektor geballt auftreten. Inwieweit die Implementierung von *Gensim* hier eine Rolle spielt, wurde nicht näher betrachtet. Laut Dokumentation[14] werden die Wortvektoren zufällig durch den Hashwert aus dem Wort zusammen mit einem Seed initialisiert. Die Abstände könnten also theoretisch größer N sein.

5.2.3 Fazit

Die Ergebnisse aller getesteten Klassifikatoren liefern ein sehr ähnliches Bild.

Keiner der Klassifikatoren unterscheidet zuverlässig zwischen korrekten und fehlerhaften Wörtern. Die Unterscheidung ist nicht einmal ansatzweise vorhanden, abgesehen von den Wörtern, die nicht im Vokabular eines Modells enthalten sind.

Damit lässt sich die erste, in dieser Arbeit gestellte Frage, siehe 1.1, zur Hälfte beantworten: Alle Daten in dieser Arbeit weisen darauf hin, dass Word2Vec-Wortvektoren in Kombination mit einfachen, binären Schwellwert-basierten Klassifikatoren nicht dazu geeignet sind, fehlerhafte Wörter in einem Text zu identifizieren.

Die zweite Hälfte der ersten Frage, ob das Word2Vec-Modell in Kombination mit Schwellwert-basierten Klassifikatoren zur Identifizierung fehlerhafter Wörter geeignet ist, kann in dieser Arbeit nicht abschließend beantwortet werden.

[14] Die Dokumentation der Word2Vec-Implementierung von *Gensim* kann unter der folgenden Adresse aufgerufen werden: `https://radimrehurek.com/gensim/models/word2vec.html` – zuletzt aufgerufen am 07. 01. 2016.

Es ist möglich, dass ein Verfahren entwickelt werden kann, welches das Word2Vec-Modell erfolgreich zur Klassifizierung fehlerhafter Wörter verwendet. Dieses könnte Schwellwert-basierte Klassifikatoren verwenden, oder gänzlich andere Klassifikatoren. Die Methoden aus dieser Arbeit eignen sich dazu nicht.

In dieser Arbeit wurden viele fehlerhafte Wörter dadurch gefunden, dass sie nicht im Vokabular des Word2Vec-Modells enthalten sind. Je nach Modell sind mehr oder weniger korrekte Wörter ebenfalls nicht im Vokabular enthalten, siehe 5.2.1 für eine ausführliche Analyse.
Folglich könnte man vom Word2Vec-Modell nur das Vokabular verwenden und eine Wörterbuchsuche darauf anwenden. Dieses Vorgehen ist jedoch nur bedingt zu empfehlen, da mehr korrekte Wörter nicht im Vokabular enthalten sind als tatsächlich fehlerhafte, siehe Abschnitt 5.2.1.

In Bezug auf die zweite Frage werden die optimalen Klassifikationsschranken in der folgenden Tabelle dargestellt. Die Schwellwerte wurden durch die Bestimmung des höchsten F1-Scores, siehe Abschnitt 4.4, für jeden Klassifikator ermittelt.

Tabelle 5.13: Die optimalen Klassifikations-Schwellwerte (*Klassifikations-Schranken*), ermittelt durch den maximalen F1-Score. Die Schwellwerte sind pro Klassifikator und jeweils pro Modell angegeben.

Klassifikator	**Wikipedia-Modell**		**Zeitungsartikel-Modell**	
	Schranke	**F1-Score**	**Schranke**	**F1-Score**
Kontext-basiert	10^{-16}	0,372737375	0,00027	0,070300405
Vorgänger-basiert	10^{-19}	0,217654414	0,00027	0,060051500
Mittelvektor	10^{-12}	0,372737375	0,00027	0,080807786
Kosinus-Abstand	0,1	0,372492674	0,1	0,092273759
Euklidische Distanz	0,65	0,372737375	0,75	0,063807088

Diese Schranken können jeweils als optimal angesehen werden, da die meisten korrekten Wörter als korrekt klassifiziert werden und gleichzeitig über 85% der fehlerhaften Wörter als solche erkannt werden. Die Precision ist bei der Wahl dieser Schwellwerte nahe oder gleich dem erreichten Maximum.

Zuletzt zur dritten, in dieser Arbeit gestellten Frage, ob domänenspezifische Wortvektoren bessere Ergebnisse liefern als domänenunabhängige Wortvektoren. Bei der Betrachtung der Wortvektoren gibt es Argumente, die für beide Modelle sprechen, wie beispielsweise beim *Mittelvektor Klassifikator*. Das Zeitungsartikel-Modell hat jedoch keine besseren Ergebnisse geliefert als das Wikipedia-Modell und umgekehrt. Bei Betrachtung des Vokabulars wird ersichtlich, dass weit weniger Wörter im Vokabular des Zeitungsartikel-Modells enthalten sind als im Vokabular des Wikipedia-Modells. Dadurch liefert das Modell mit dem domänenunabhängigen Trainingskorpus bessere Ergebnisse als das Modell mit dem domänenspezifischen Trainingskorpus. Der Unterschied in der Anzahl der Wörter dürfte u.a. im Umfang der Trainingskorpora begründet sein. Der domänenspezifische Trainingskorpus beinhaltet etwa $\frac{1}{1000}$ der Wörter des domänenunabhängigen Trainingskorpus.
Für die in dieser Arbeit verwendeten und untersuchten Daten ist das Modell mit dem domänenunabhängigen Trainingskorpus klar vorzuziehen.

6 Zusammenfassung

Ziel dieser Arbeit war, Word2Vec zusammen mit einfachen Schwellwert-basierten Klassifikatoren zur Fehlererkennung in mit OCR digitalisierten Texten anzuwenden. Indem die Fehler erkannt werden, sollte der manuelle Aufwand zum Durchlesen und Korrigieren der Texte reduziert werden können.

Es wurde dargelegt, welche Fehler bei der Digitalisierung von Texten mit OCR vorkommen können. Als Grundlage dienten vor allem historische Zeitungsartikel in deutscher Sprache aus der Zeit des ersten Weltkriegs.

Es wurden mehrere einfache Schwellwert-basierte Klassifikatoren vorgestellt und getestet. Die Klassifikatoren arbeiten mit der CBOW-Variante von Word2Vec und lassen sich in zwei Gruppen einteilen:
Die erste Gruppe arbeitet mit Methodiken, die sich an die Methoden zur Berechnung der Wahrscheinlichkeiten beim CBOW-Modell anlehnen. Die zweite Gruppe arbeitet nur mit den CBOW-Wortvektoren.

Zur Evaluierung der Klassifikatoren wurden mehrere Experimente unter gleichen Rahmenbedingungen und mit unterschiedlicher Parametrisierung durchgeführt. Es wurden zwei CBOW-Modelle berechnet: Ein Modell mit einem domänenspezifischen Trainingskorpus und ein Modell mit einem domänenunabhängigen Trainingskorpus. Als Validierungsdaten wurde eine Auswahl historischer, mit OCR digitalisierter Zeitungsartikel aus dem beginnenden 20. Jahrhundert verwendet.

Mit der Evaluierung sollten 3 Fragen beantwortet werden:
1. Eignen sich das Word2Vec-Modell und die Word2Vec-Wortvektoren für eine Schwellwert-basierte Fehlererkennung?
Diese Frage konnte in dieser Arbeit nicht abschließend geklärt werden. Die evaluierten Klassifikatoren eignen sich nicht zur Identifizierung fehlerhafter Wörter. Durch eine Wörterbuchsuche im Vokabular des CBOW-Modells konnten jedoch viele fehlerhafte Wörter gefunden werden. Dennoch ist die Precision nicht über 0,25 gestiegen, was nur bedingt für eine Wörterbuchsuche spricht.
2. Wie verhalten sich unterschiedliche Schwellwert-basierte Klassifikatoren bei verschiedenen Schwellwerten?
Hier sollte untersucht werden, ob es bei den einzelnen Klassifikatoren einen optimalen Schwellwert zur Fehlererkennung gibt. Diese Schwellwerte wurde durch den maximal erreichten F1-Score ermittelt und sind in Tabelle 5.13 angegeben.
3. Lassen sich mit domänenspezifischen Wortvektoren bessere Ergebnisse erzielen als mit domänenunabhängigen Wortvektoren?
Für diese Arbeit lieferte das CBOW-Modell, das mit einem domänenunabhängigen Textkorpus trainiert wurde, bessere Ergebnisse. Dies liegt vor allem in der unterschiedlichen Vokabulargröße der CBOW-Modelle begründet. Abgesehen davon konnte kein großer Unterschied bei der Verwendung von domänenspezifischen und domänenunabhängigen Wortvektoren festgestellt werden.

Literaturverzeichnis

[1] K. Kukich, "Techniques for automatically correcting words in text," *ACM Computing Surveys (CSUR)*, vol. 24, no. 4, pp. 377–439, 1992.

[2] F. Ahmed, E. W. De Luca, and A. Nürnberger, "Revised n-gram based automatic spelling correction tool to improve retrieval effectiveness," *Polibits*, vol. 40, pp. 39–48, 2009.

[3] A. Hassan, S. Noeman, and H. Hassan, "Language independent text correction using finite state automata.," in *IJCNLP*, pp. 913–918, 2008.

[4] M. Richter, P. StraÅák, and A. Rosen, "Korektor-a system for contextual spellchecking and diacritics completion.," in *COLING (Posters)*, pp. 1019–1028, 2012.

[5] P. Samanta and B. B. Chaudhuri, "A simple real-word error detection and correction using local word bigram and trigram.," in *ROCLING*, 2013.

[6] C. Whitelaw, B. Hutchinson, G. Y. Chung, and G. Ellis, "Using the web for language independent spellchecking and autocorrection," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pp. 890–899, Association for Computational Linguistics, 2009.

[7] A. R. Golding and Y. Schabes, "Combining trigram-based and feature-based methods for context-sensitive spelling correction," in *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pp. 71–78, Association for Computational Linguistics, 1996.

[8] E. Bick, "A constraint grammar based spellchecker for danish with a special focus on dyslectics," 2006.

[9] M. Hermet, A. Désilets, and S. Szpakowicz, "Using the web as a linguistic resource to automatically correct lexico-syntactic errors," 2008.

[10] G. Hirst and A. Budanitsky, "Correcting real-word spelling errors by restoring lexical cohesion," *Natural Language Engineering*, vol. 11, no. 01, pp. 87–111, 2005.

[11] A. R. Golding, "A bayesian hybrid method for context-sensitive spelling correction," *arXiv preprint cmp-lg/9606001*, 1996.

[12] A. R. Golding and D. Roth, "A winnow-based approach to context-sensitive spelling correction," *Machine learning*, vol. 34, no. 1-3, pp. 107–130, 1999.

[13] J. Sjöbergh, "Faking errors to avoid making errors: Machine learning for error detection in writing," 2005.

[14] M. P. Jones and J. H. Martin, "Contextual spelling correction using latent semantic analysis," in *Proceedings of the fifth conference on Applied natural language processing*, pp. 166–173, Association for Computational Linguistics, 1997.

[15] E. Brill and R. C. Moore, "An improved error model for noisy channel spelling correction," in *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pp. 286–293, Association for Computational Linguistics, 2000.

[16] R. Holley, "How good can it get? analysing and improving ocr accuracy in large scale historic newspaper digitisation programs," *D-Lib Magazine*, vol. 15, no. 3/4, 2009.

[17] E. Klijn, "The current state-of-art in newspaper digitization: A market perspective," *D-Lib Magazine*, vol. 14, no. 1, p. 5, 2008.

[18] E. M. Riseman and A. R. Hanson, "A contextual postprocessing system for error correction using binary n-grams," *Computers, IEEE Transactions on*, vol. 100, no. 5, pp. 480–493, 1974.

[19] X. Tong and D. A. Evans, "A statistical approach to automatic ocr error correction in context," in *Proceedings of the fourth workshop on very large corpora*, pp. 88–100, 1996.

[20] L. Eikvil, "Optical character recognition," 1993.

[21] H. Niwa, K. Kayashima, and Y. Shimeki, "Postprocessing for character recognition using keyword information.," in *MVA*, pp. 519–522, 1992.

[22] X. Tong, C. Zhai, N. Milic-Frayling, and D. A. Evans, "Ocr correction and query expansion for retrieval on ocr data–clarit trec-5 confusion track report.," in *TREC*, 1996.

[23] J. J. Hull, "Incorporating language syntax in visual text recognition with a statistical model," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 18, no. 12, pp. 1251–1255, 1996.

[24] I. Guyon and F. Pereira, "Design of a linguistic postprocessor using variable memory length markov models," in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, vol. 1, pp. 454–457, IEEE, 1995.

[25] T. M. Breuel, "The ocropus open source ocr system," in *Electronic Imaging 2008*, pp. 68150F–68150F, International Society for Optics and Photonics, 2008.

[26] T. M. Breuel, A. Ul-Hasan, M. A. Al-Azawi, and F. Shafait, "High-performance ocr for printed english and fraktur using lstm networks," in *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pp. 683–687, IEEE, 2013.

[27] Y. Bengio, Y. LeCun, and D. Henderson, "Globally trained handwritten word recognizer using spatial representation, convolutional neural networks, and hidden markov models," *Advances in Neural Information Processing Systems*, pp. 937–937, 1994.

[28] J. Turian, L. Ratinov, and Y. Bengio, "Word representations: a simple and general method for semi-supervised learning," in *Proceedings of the 48th annual meeting of the association for computational linguistics*, pp. 384–394, Association for Computational Linguistics, 2010.

[29] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *The Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[30] Y. Bengio, "Neural net language models," *Scholarpedia*, vol. 3, no. 1, p. 3881, 2008.

[31] P. D. Turney, P. Pantel, *et al.*, "From frequency to meaning: Vector space models of semantics," *Journal of artificial intelligence research*, vol. 37, no. 1, pp. 141–188, 2010.

[32] P. D. Turney and M. L. Littman, "Measuring praise and criticism: Inference of semantic orientation from association," *ACM Transactions on Information Systems (TOIS)*, vol. 21, no. 4, pp. 315–346, 2003.

[33] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *The Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[34] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[35] D. R. G. H. R. Williams and G. Hinton, "Learning representations by back-propagating errors," *Nature*, pp. 323–533, 1986.

[36] G. E. Hinton, "Learning distributed representations of concepts," in *Proceedings of the eighth annual conference of the cognitive science society*, vol. 1, p. 12, Amherst, MA, 1986.

[37] G. E. Hinton, "Connectionist learning procedures," *Artificial intelligence*, vol. 40, no. 1, pp. 185–234, 1989.

[38] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[39] X. Rong, "word2vec parameter learning explained," *arXiv preprint arXiv:1411.2738*, 2014.

[40] L. Wolf, Y. Hanani, K. Bar, and N. Dershowitz, "Joint word2vec networks for bilingual semantic representations," *International Journal of Computational Linguistics and Applications*, vol. 5, no. 1, pp. 27–44, 2014.

[41] R. Banjade, N. B. Niraula, N. Maharjan, V. Rus, D. Stefanescu, M. Lintean, and D. Gautam, "Nerosim: A system for measuring and interpreting semantic textual similarity,"

[42] W. Ling, C. Dyer, A. Black, and I. Trancoso, "Two/too simple adaptations of word2vec for syntax problems," *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL), Denver, CO*, 2015.

[43] Y.-R. Wang and Y.-F. Liao, "Word vector/conditional random field-based chinese spelling error detection for sighan-2015 evaluation," *ACL-IJCNLP 2015*, p. 46, 2015.

[44] Y.-R. Wang, Y.-F. Liao, Y.-K. Wu, and L.-C. Chang, "Conditional random field-based parser and language model for traditional chinese spelling checker," in *Proceedings of the 7th SIGHAN Workshop on Chinese Language Processing (SIGHAN?13)*, pp. 69–73, 2013.

Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Passau, den 7. Januar 2016

Maria Kober